AFOSR-TR-97-0678

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
1 December 1996

3. REPORT TYPE AND DATES COVERED
Final Technical Report

4. TITLE AND SUBTITLE
Fusion of Sensors That Interact Dynamically for Exploratory Development of Robust, Fast Object Detection and Recognition

5. FUNDING NUMBERS
Contract No.:
F49620-95-C-0072

6. AUTHOR(S)
Cesar Bandera, Jing Peng

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Amherst Systems Inc.
30 Wilson Road
Buffalo, NY 14221

8. PERFORMING ORGANIZATION REPORT NUMBER
621-9160006

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, TX 77058

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
0002AA

11. SUPPLEMENTAL NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release;
distribution unlimited.

12a. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Foveal vision simultaneously offers a wide field-of-view for better detection and central high acuity for better recognition; additionally, it is highly optimized and cost-effective for time-critical active vision applications. However, space-variant data acquisition necessitates the development of gaze control techniques for demand-driven allocation of resources to improve relevant information acquisition and the overall system performance. This report describes a commercially feasible, efficient reinforcement learning approach to gaze control for foveal machine vision. The report first lays a theoretical foundation for reinforcement learning. It then introduces particular reinforcement learning algorithms in conjunction with function approximation as an efficient learning control method for visual attention. The efficacy of the method is validated on a number of moderately complex target detection and active perception problems. By contrast, the substantial body of work on gaze control for active vision has not taken advantage of the power and flexibility of machine learning methods for visual attention. Finally, the report describes several experiments designed to evaluate the relative efficiency of various reinforcement learning algorithms and techniques for input generalization using both prediction and control problems. Computational results show that reinforcement learning with neuro function approximation can be successfully used to obtain achievable gaze control performance in commercially feasible foveal machine vision products.

14. SUBJECT TERMS
Reinforcement Learning, Active Vision, Hierarchical Processing, Foveal Vision, Multiresolution

15. NUMBER OF PAGES
38

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
Unlimited

AMHERST SYSTEMS INC.

FUSION OF SENSORS THAT INTERACT DYNAMICALLY FOR EXPLORATORY DEVELOPMENT OF ROBUST, FAST OBJECT DETECTION AND RECOGNITION

FINAL REPORT

AFOSR STTR Phase I contract

Contract No. F49620-95-C-0072

December 1, 1996

Document Control No.: 621-9160006

Final Report

Prepared for:

Air Force Office of Scientific Research
110 Duncan Avenue, Suite B115
Bolling AFB, DC 20332-0001

Prepared by:

Amherst Systems Inc.
Buffalo, NY 14221-7082
(716) 631-0610

PI: Dr. Cesar Bandera
Dr. Jing Peng
{bandera,jpe}@amherst.com

AMHERST SYSTEMS INC.

FUSION OF SENSORS THAT INTERACT DYNAMICALLY FOR EXPLORATORY DEVELOPMENT OF ROBUST, FAST OBJECT DETECTION AND RECOGNITION

FINAL REPORT

December 1, 1996

Prepared for: Air Force Office of Scientific Research
110 Duncan Avenue, Suite B115
Bolling AFB, Wash, DC 20332-0001

Contract No.: F49620-95-C-0072

CDRL Item: 0002AA

Cesar Bandera
Program Manager

Daniel G. Sentz
Quality Assurance Manager

Abstract

Foveal vision simultaneously offers a wide field-of-view for better detection and central high acuity for better recognition; additionally, it is highly optimized and cost-effective for time-critical active vision applications. However, space-variant data acquisition necessitates the development of gaze control techniques for demand-driven allocation of resources to improve relevant information acquisition and the overall system performance. This report describes a commercially feasible, efficient reinforcement learning approach to gaze control for foveal machine vision. The report first lays a theoretical foundation for reinforcement learning. It then introduces particular reinforcement learning algorithms in conjunction with function approximation as an efficient learning control method for visual attention. The efficacy of the method is validated on a number of moderately complex target detection and active perception problems. By contrast, the substantial body of work on gaze control for active vision has not taken advantage of the power and flexibility of machine learning methods for visual attention. Finally, the report describes several experiments designed to evaluate the relative efficiency of various reinforcement learning algorithms and techniques for input generalization using both prediction and control problems. Computational results show that reinforcement learning with neuro function approximation can be successfully used to obtain achievable gaze control performance in commercially feasible foveal machine vision products.

Section 1: INTRODUCTION

Real-time machine vision is characterized by an overwhelming data rate, typically measured in pixels of uniform resolution imagery per second. This data rate imposes severe performance constraints on machine vision that have limited its application to simple tasks in rigorously controlled environments, and have precluded smart vision for autonomous platforms in dynamic, real-world scenarios. For example, the robustness of image processing and the degree of image understanding is limited because there are typically insufficient resources (time, hardware, communications bandwidth, etc.) to execute necessary algorithms; the acquisition of visual information is limited by the trade-off between field-of-view (FOV), spatial resolution (i.e., pixels per degree), and temporal resolution (i.e., frame rate). Possibly the most inefficient aspect of current machine vision, such as automatic target recognition (ATR) and visual trackers, is inherent to the measurement process itself. Current machine vision imaging sensors use uniform sampling within the FOV. Consequently, all measurements are made with the same spatial resolution. However, scene features relevant to the platform mission are typically localized within the FOV, e.g., small strategic relocatable targets hidden within a large area being searched by an ATR system, or unresolved maneuvering targets being tracked within a wide corridor. In such cases, uniformly sampling within the FOV is inappropriate; regions with little or no relevance to the mission are sampled at the same resolution as key features. The resulting irrelevant data wastefully increases the system data rate because it occupies valuable signal bandwidth and computational/data storage resources without adding (relevant) information.

In contrast to the uniform acuity of conventional machine vision, virtually all advanced biological vision systems sample the scene in a space-variant fashion. Retinal acuity varies by several orders of magnitude within the FOV. The region of the retina with notably high acuity, called the fovea, is typically a small percentage of the overall FOV (≈5%) centered at the optical axis. The wide FOV, supported by lower peripheral acuity, and the high acuity fovea impose a much smaller data rate than supporting the entire FOV uniformly at high acuity. Inherent with space-variant sampling is the context-sensitive articulation of the sensor's optical axis, whereby the fovea is aligned with relevant features in the scene. These features can be targets or classification features on the targets themselves. Space-variant sampling and gaze control are collectively called *foveal vision.*

The key advantages of foveal vision are its simultaneous wide FOV, high resolution, and fast frame rates. This makes foveal vision particularly well suited for time critical applications with payload constraints, such as those of fast automata and active pursuit scenarios. Foveal vision also is well suited for applications in which (1) scenario conditions cannot be controlled well or anticipated, or (2) the system must simultaneously perform very different tasks, such as target tracking, object recognition, or navigation. It is typically unfeasible to build a uniform acuity vision system that can efficiently perform all these tasks in all these conditions. For example, the combination of high resolution, wide FOV, and rapid frame updates would overload a uniform acuity system (the bottleneck would start at the sensor read-out mechanism).

All the benefits of foveal machine vision are contingent on gaze control performance. There are two general classes of gazing. The first kind, initiated by low-level functions, is a pre-attentive behavior; it is a fast "reflex" response driven by stimuli prior to their assimilation into high-level system perception. The second kind, initiated by high-level functions, is an attentive behavior; it is a planned response driven by system perception. The combination of pre-attentive and attentive behaviors permits foveal vision to execute planned operations, such as target search and discriminating interrogation, while responding rapidly to the unpredictable events. This report is concerned with efficient learning approaches to gazing control of the second kind.

The technique we introduce here is a combination of reinforcement learning and biologically inspired function approximators such as neural networks, both of which have been used successfully in a wide variety of tasks ranging from predictions to nonlinear dynamic modeling. In particular, reinforcement learning by itself can produce useful behaviors without higher level command (e.g., semantics). More specifically, we describe efficient gaze control algorithms that combine Q(λ)-like learning [PengWilliams94,96] or State-Action-Reward-State-Action (SARSA) [RummeryNiranjan94] techniques with parameterized function approximators, such as the Cerebellar Model Articulation Controller (CMACs) [Albus75, MillerGlanzKraft87] and radial basis functions (RBFs) [Powell87]. While reinforcement learning addresses the problem of improving performance as evaluated by *any* measure whose values can be supplied to the learning system, general approximators such as CMACs are compact representation schemes that are capable of implementing any function to any desired degree of accuracy given sufficient resources. The key benefits of the proposed technique are its ability to improve plant performance over time in less structured environments, its robustness and learning efficiency, and its scalability to large-scale nonlinear problems. In addition, it is feasible to construct fast, parallel devices to implement this technique for real-time applications. Such abilities are the prerequisite for any system to operate in real world conditions. Furthermore, such algorithms, when combined with a plant model, may significantly reduce the overall cost of the development of gaze control strategies.

1.1 Why Learning?

Gaze control for foveal machine vision must maintain closed-loop system integrity and performance over a wide range of operating conditions. This objective is difficult to achieve due to several circumstances including the complexity of both the plant and the performance objectives, due to the presence of uncertainty, and most of all due to extremely limited *a priori* model information. Under such conditions, it is very difficult or even impossible to design a control policy with fixed properties that meets the desired performance specifications. From a design standpoint, these difficulties may result from nonlinear or time-varying behavior, poorly modeled environment and plant dynamics, high dimensionality, multiple inputs and outputs, complex objective functions, constraints, and the possibility of actuator or sensor failures. Each of these effects, if present, must be addressed if the system is to operate properly in an autonomous or semi-autonomous fashion for extended periods of time.

The system designer may take several actions: (1) reduce the desired level of performance, (2) conduct additional theoretical or empirical model development in advance to reduce uncertainty, or (3) design the control system to adjust itself on-line automatically to reduce uncertainty and/or improve performance. The third action is significantly different from the first two because the resulting design is not fixed, and instead has inherent operational flexibility.

Learning control techniques are intended for those difficult situations where it is unacceptable, on the basis of requirements, cost, or feasibility, to reduce the desired level of closed-loop system performance, or to undertake additional model development and validation. As a consequence, learning control increases the level of achievable performance by reducing uncertainty on-line through direct interaction with the actual system and by effective use of experience. Learning control therefore offers several unique features that make it attractive for building foveal vision systems that must operate autonomously in uncertain environments.

It is important to realize that many learning tasks arising in control require methods that cannot be precisely characterized as supervised learning. Imagine one wishes to adjust a control law in order to improve the performance of a plant as measured by a performance criterion that evaluates the overall behavior of the plant; for example, one might wish to minimize a measure of the plant's energy cost over time. Optimal control methods apply if models exist of the plant and performance measure that are sufficiently accurate and tractable. However, in less structured situations it is possible to improve plant performance over time by means of on-line methods performing what is called *reinforcement learning*. Whereas the performance measure for supervised learning is defined in terms of a set of desired targets by means of a known error criterion such as mean-squared error, reinforcement learning is concerned with the problem of learning from rewards and punishments (see below for a detailed discussion). Therefore, in tasks of this kind there exist desired control signals– those that lead to optimal plant performance– but the learning system is not told what they are because there is no teacher available. The problem is to find these optimal control signals, not simply to remember and generalize from them. It is this kind of learning control problem that this effort addresses.

Note that both adaptive and learning control systems are capable of automatically adjusting their internal representations on-line, achieved either directly by changing the control law itself, or indirectly through model identification and control law redesign. Also, both make use of performance feedback information gained through closed-loop interactions with the plant and its environment. However, the major differences are a matter of degree, emphasis, and intended purpose. A control system that treats every distinct operating situation as novel is limited to adaptive operation, whereas a system that correlates past experiences with past situations, and that can recall and exploit those experiences, is capable of learning. Since, in the process of learning, the learning system must be capable of adjusting its memory to accommodate new experiences, a learning system must in some sense incorporate an adaptive capability. However, the design and intended purpose of a learning system require capabilities beyond that of adaptation. This report is concerned with learning control systems.

The overall technological innovation research into gaze control for active foveal machine vision was divided between the Amherst Systems team and the UB team. The Amherst Systems team took the lead in addressing gaze control issues that are related to foveal target recognition, whereas the UB team took the lead in addressing issues that arise in foveal target search and detection. The resulting technical innovations from both efforts are essential for developing commercially feasible foveal machines for ATR.

The body of the report is organized as follows. Section 2 gives a brief introduction to machine learning in general and reinforcement learning in particular, including issues associated with reinforcement learning. Section 3 introduces a mathematical framework, namely *dynamic programming*, from which various reinforcement learning techniques are derived. Section 4 describes an experiment designed to evaluate reinforcement learning in conjunction with recurrent neural networks to gaze control for active perception. Section 5 describes UB's effort in developing efficient reinforcement learning methods for foveal target detection. Section 6 describes several experiments comparing the relative efficiency of various reinforcement learning algorithms using both prediction and control tasks.. Finally, Section 7 summaries the research effort targeted at science to technology transfer.

Section 2: REINFORCEMENT LEARNING AND GENERAL FUNCTION APPROXIMATORS

This section provides a brief overview of reinforcement learning and general function approximation schemes. A more thorough analysis of the subject is presented in [Peng94, Girosi95].

2.1 Reinforcement Learning

Learning is defined in this report as the capability of any system to change itself over time with the intent of improving performance on tasks defined by its environment. Most learning problems can be divided into three broad categories: *unsupervised*, *supervised*, and *reinforcement*s learning problems. Unsupervised learning is concerned with clustering input data into categories that can later be used to predict future data. It receives no feedback from outside and is the environment that provides an implicit feedback. In contrast, supervised learning uses feedback provided by an external teacher to learn input-output mappings. In supervised learning, the system receives an input instance, produces an output, receives the desired output from the teacher, and then uses the discrepancy between its actual output and the desired output to make an adjustment to itself. Reinforcement learning is similar to supervised learning in that it receives an outside feedback. However, the nature of the feedback is different. The feedback is instructive in the case of supervised learning and evaluative in the case of reinforcement learning. That is, for supervised learning the system is presented with the correct output for each input instance, while for reinforcement learning the system produces a response that is then evaluated using a scalar indicating the appropriateness of the response. As an example, a checker playing program that uses the outcome of a game to improve its performance is a reinforcement learning system. Knowledge about an outcome is useful for evaluating the total system's performance, but it says nothing about which actions were instrumental for the ultimate win or loss. Learning in pattern classification is not reinforcement learning since during the training such systems receive information from a teacher about the correct classification. In general, reinforcement learning is more widely applicable than supervised learning since any supervised learning problem can be treated as a reinforcement learning problem. In the context of building learning control systems whose behaviors are to be developed autonomously, reinforcement learning provides an attractive framework because of its similarity to the problem faced by animals that are expected to behave intelligently in situations with great uncertainty. This type of learning is also used to model behavior learning in animals and humans in experimental psychology, and is the main focus of this Phase I effort.

2.2 Associative versus Non-associative Learning

One variation in reinforcement learning is the distinction between associative and non-associative learning. In the non-associative reinforcement learning paradigm, reinforcement is the only information the learning system receives from its environment. That is, the system has no input information about the environment; it does not know the state of the environment in which it is embedded. In these tasks, the objective of the system is to determine the optimal action that maximizes its expected reinforcement. Non-associative reinforcement learning does not concern us here because our applications require that we take into account the changes of environmental states. Non-associative reinforcement learning has been mainly studied in the context of function optimization [WilliamsPeng93] and learning automata theory [NarendraThathatchar89].

Associative reinforcement learning extends non-associative reinforcement learning in a natural way. In such a learning paradigm, the control system receives input information that indicates the state of its environment as well as reinforcement. The additional information tells what state of its environment the system is in. It should be pointed out that, in general, input the system receives about its environment is usually incomplete in that the input only reveals partial information about the state of its environment due to the system's limited sensory capabilities. To emphasize that input values need not be in a one-to-one correspondence with environmental states, the term *sensations* is often used for these input values.

The objective for the system in these tasks is to learn an optimal mapping from the sensations about its environment to the actions it can execute. In the simplest case, the mapping will be a function of the current sensation. In general, however, the system might try to represent states that summarize all of the past sensations. Note that, at each state, the job of associative reinforcement learning is the same as that of non-associative reinforcement learning. Associative reinforcement learning is of great interest to us here due to its similarity to modeling the learning and decision-making tasks faced by animals and humans.

2.3 Immediate versus Delayed Reinforcement Learning

Another complication to reinforcement learning is the timing of reinforcement. In the simplest tasks, the system receives, after each action, reinforcement indicating the goodness of that action. Immediate reinforcement occurs commonly in non-associative learning tasks. In more complex tasks, however, reinforcement is often temporally delayed, occurring only after the execution of a sequence of actions. Delayed reinforcement learning is important because in many domains, immediate reinforcement regarding the value of a decision may not always be available. For example, in a chess game, the outcome of the game is not known until the game is over. The difficulty associated with delayed reinforcement learning is that some judicious actions now may only allow high rewards to be achieved later; each action in an action sequence may be essential to achieving a reward, even though not all of the actions are followed by immediate rewards. Conversely, in a chess game, a move may lead a particular player into a "doomed" situation from which it is impossible to prevent that player from eventually losing the game– but the loss actually occurs some time later. It would be unwise to blame the actions taken immediately before the game was lost, for these actions may have been the best that could be taken in the circumstances. The actual mistake may have been made some time earlier. The problem of assigning credit or blame to decisions when reinforcement is delayed is the well-known *credit assignment problem* [Minsky61]. This Phase I effort focuses on methods for efficient learning in delayed reinforcement settings.

2.4 Exploitation versus Exploration

Another important aspect of reinforcement learning is that a learning system needs two capabilities if it is to act optimally in its environment with reasonable efficiency: exploitation and exploration. Trying to satisfy both goals at the same time leads to a conflict. Exploitation suggests that the scope of the search should be narrowed as experience about actions accumulates, when an exhaustive search is not feasible, or when the system has only a finite life span, so that the system will be able to concentrate on performing actions that are expected to be good in order to increase overall rewards; exploration suggests that the scope of the search should not be narrowed irrevocably so far as to let superior actions with little experience slip through permanently.

The tradeoff most reinforcement learning systems make to resolve the conflict is to occasionally take random actions. This ad-hoc approach works well for most problems encountered empirically, but has no strong theoretical foundation. A more statistically sound exploration strategy, the *Interval Estimation* (IE) algorithm [Kaelbling90], has been proposed that uses the combination of mean and confidence to choose appropriate actions. Other approaches have also been suggested [Sutton90].

2.5 Direct versus Indirect Learning

There are two general approaches to learning optimal policies. One is the *indirect* or *model-based* approach, which requires first learning an environmental model in the form of state-transition and reward probabilities. A search or computational technique such as dynamic programming is then applied to produce an optimal policy for the system. Most work in the adaptive control of Markov processes described in the control engineering literature has been model-based. Learning a world model is a supervised learning that is relatively simple compared to the expensive search for an optimal policy or a value function. It is interesting to note that a learned world model need not be completely accurate for dynamic programming to produce a policy that accomplishes underlying tasks. Very often an approximate model is sufficient.

The second approach to learning how to obtain optimal policies is *direct* [BartoSuttonWatkins90] or *primitive* learning [Watkins89]. Instead of learning an environmental model, a direct method adjusts the policy on the basis of its observed consequences. The system must act in its environment, try a variety of actions, observe their consequences, and adjust its policy to improve performance over time. This process is known as reinforcement learning.

It is possible to combine direct methods with indirect methods in a variety of ways. Sutton's Dyna architecture is a classic example [Sutton90]. Also see [MooreAtkeson92, PengWilliams93]. One typical characteristic of real world problems is their high dimensionality.

It would be impractical for a computational method, such as dynamic programming, to produce a complete solution before committing the very first act. In such large-scale problems, a learning method with generalized capabilities that computes good solutions quickly in certain important areas of the state space may be sufficient. Tesauro's TD Gammon [Tesauro92] suggests that direct methods appear to be good candidates for accomplishing this. This, however, does not imply that the advantages of model-based learning should be overlooked. In fact, in the context of learning, the combinations of direct learning and model-based learning can offer significant performance improvements over either technique alone and make learning systems most promising.

2.6 Q(λ) Learning

The Q(λ) learning algorithm [PengWilliams94,96] is an incremental multi-step method that extends the one-step Q-learning algorithm [Watkins89] by combining it with TD(λ) returns ($0 \leq \lambda \leq 1$) [Sutton88] for learning from delayed rewards. λ is a weighting parameter that determines to what extent future rewards will contribute to the estimates of current action values. By allowing corrections to be made incrementally to the predictions of observations occurring in the past, the Q(λ) learning method propagates information rapidly to where it is important. The Q(λ) learning algorithm works significantly better than the one-step Q-learning algorithm on a number of tasks, and its basis in the integration of one-step Q-learning and TD(λ) returns makes it possible to take advantage of some of the best features of both the Q-learning and actor-critic learning paradigms and to be a potential bridge between them. It can also serve as a basis for developing various multiple time-scale learning mechanisms that are essential for applications of reinforcement learning to large-scale problems.

2.7 Function Approximators

Global function approximations assume that there is an underlying, learnable structure in the problem. This assumption is essential in order for a generalization to make sense. Some approximation methods, such as polynomial approximations and neural networks [Barnhill77, Franke82, Schumaker76], use a strong form of the assumption. Global models that are sufficient for capturing the structure of the problem being modeled are usually used to fit entire training data. The difficulty is to find an appropriate structure for a global model. One simply cannot tell if a given set of parameters would be sufficient to produce an adequate model for the data, unless it is chosen with *a priori* knowledge. An additional difficulty associated with connectionist networks is that because an iterative gradient descent search procedure is used to find the appropriate set of weights, training data must be repeatedly presented to the network. In the case of on-line learning, this implies that old experiences have to be stored and presented again since learning new experiences may degrade the representation of older experiences. Other methods, such as decision trees and the nearest neighbor generalization, use a weaker form of the assumption.

Sparse coarse coded function approximators, such as the CMAC and RBFs, have the same representational capability as neural networks; however, they are more efficient computationally and work well in many other respects such as better response to spatial variations. In addition, basis functions decrease with the distance of the data points from the evaluation point, so that only the neighboring points affect the estimate of the function at the evaluation point, therefore providing a "local" approximation scheme. It should be noted, however, that these techniques in general do not have the highest possible degree of locality since the parameter that controls the locality is the same for all the basis functions. It is possible to design even more local techniques, in which each basis has a parameter that controls its locality. The CMAC and RBFs are the main focus of this Phase I effort.

Section 3: FOUNDATION OF REINFORCEMENT LEARNING

Dynamic programming provides solutions to problems of credit assignment in delayed reinforcement learning and optimal control. The significance of dynamic programming has often been under-appreciated in artificial intelligence (AI) research. The appeal and large potential of dynamic programming in AI are due to the fact that it provides an attractive basis for compiling actions into reactive policies that can be used for real-time control, as well as for developing methods for learning such policies when the world model is not available. It should be emphasized at the outset that although dynamic programming requires a complete model of the world, it is the framework it provides that is relevant to the synthesis and analysis of a variety of reinforcement learning algorithms. Dynamic programming also has been used in behavioral ecology for the study of animal behavior.

The fundamental principle of dynamic programming is to determine an optimal policy by constructing a *value function* or a *return function* on the state space. This chapter discusses dynamic programming in the context of Markov decision processes and introduces the major difficulties faced by dynamic programming. The discussion is based on [Peng94].

3.1 Stochastic Dynamic Programming

One of the most powerful techniques developed for the solution of optimization problems, such as routing and scheduling, is dynamic programming (DP) [Bellman61]. It can be formulated as follows.

A *discrete-time Markov decision process* is a 4—tuple *(X,A,P,R)*, where

1. *X* is a finite set of states on which the process evolves.

2. *A* is a finite set of actions. For each state $x \in X$ there is an associated non-empty subset of *A*, whose elements are the *admissible actions* when the process is in state *x*.

3. *P* is a probabilistic state-transition law. It is a function: $X \times X \times A \to \Re$, and $P_{xy}(a)$ is the probability to reach state *y* from state *x* when action *a* is taken in that state. For any pair *(x,a)* $\sum_y P_{xy}(a) = 1$.

4. R: $X \times A \rightarrow \Re$ is the expected short-term reward function on states and actions.

A policy is in general any set of rules for selecting actions. In this discussion, we consider only policies that are non-randomized and stationary.

A *stationary policy* π is a function: $X \rightarrow A$ mapping the state space into the action space.

We use $\pi(x)$ to denote the action the policy chooses when in state x. Broadly speaking, the objective of the agent in the decision problem is to maximize the rewards it receives over time. The agent's current action influences not only the immediate reward it receives in the current state, but also the rewards it will be able to receive in the future.

There are several ways in which future rewards can be assessed: average rewards, total rewards, and total discounted rewards. In this report, the agent seeks to maximize *total discounted rewards*. This is the simplest case. Nevertheless, the learning methods can be used for learning to maximize the total reward or the average reward under certain conditions.

The total amount of discounted reward the agent will receive from time *t* is as follows:

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n} + \cdots, \qquad \text{(eq. 3.1-1)}$$

where r_i is the reward received at time i with $R(x_t, a_t) = E[r_t | x_t, a_t]$, and γ: $0 \leq \gamma < 1$, is a discount factor. The effect of the discount factor is twofold. The discount factor ensures that the quantity of (eq. 3.1-1) is well defined and a reward to be received in the future will be considered less valuable than one received now. This total discounted reward is called the *return*.

In a Markov decision process, the total return depends not only on the state the agent is in at time *t* and the policy the agent employs, but also on random factors influencing the state transitions and the rewards. The expected return, however, depends only on the current state and the policy that will be followed. This expected return is written as:

$$V^\pi(x) = E_\pi\left[\sum_{t=0}^{\infty} R(x_t, \pi(x_t))\gamma^t | x_0 = x\right]$$

where E_π indicates that the expected value is taken, given that policy π is used to choose actions. We call V^π the *value function* for policy π.

Objective: Find a policy that maximizes the expected return from every state.

The dynamic programming solution to the above problem is obtained by using a recursive functional equation that determines the optimal policy for any state at all stages. This equation is a direct consequence of Bellman's principle of optimality [Bellman61]. It can be derived for the deterministic case as follows.

$$V^*(x) = \max_{\substack{a_t \\ t=0,\cdots,\infty}} \sum_{t=0}^{\infty} R(x_t, a_t)\gamma^t,$$

where $x_0 = x$. Then

$$V^*(x) = \max_{a_0} \max_{\substack{a_t \\ t=0,\cdots,\infty}} \left\{ R(x_0, a_0) + \sum_{t=1}^{\infty} R(x_t, a_t)\gamma^t \right\} \qquad \text{(eq. 3.1-2)}$$

The first term in brackets in (eq. 2) does not depend on the second maximization. Thus, (eq. 3.1-2) becomes

$$V^*(x) = \max_{a_0} \left\{ R(x_0, a_0) + \max_{\substack{a_t \\ t=1,\cdots,\infty}} \left[\sum_{t=1}^{\infty} R(x_t, a_t)\gamma^t \right] \right\}$$

$$= \max_{a_0} \left\{ R(x_0, a_0) + \gamma \max_{\substack{a_{t+1} \\ t=0,\cdots,\infty}} \sum_{t=0}^{\infty} R(x_{t+1}, a_{t+1})\gamma^t \right\}$$

$$= \max_{a_0} \left\{ R(x_0, a_0) + \gamma V^*(y) \right\} \qquad \text{(eq. 3.1-3)}$$

where y is the actual next state when taking action a_0 that maximizes (eq. 3.1-3) in state x. In general, however, the transition to a next state y occurs with probability $P_{xy}(a)$. Thus, for a Markov decision process, we have (see [Ross83])

$$V^*(x) = \max_{a} \left\{ R(x, a) + \gamma \sum_{y} P_{xy}(a) V^*(y) \right\} \qquad \text{(eq. 3.1-4)}$$

This equation is a mathematical form of Bellman's principle of optimality. It states that the maximum expected discounted return for state x is found by taking the action that maximizes the sum of the expected reward to be received at the present state and the expected maximum discounted return from the state that results from applying the action. We call V^* the *optimal* value function and any policy whose value function is optimal an *optimal* policy π^*. It is the optimal value function that the agent tries to approximate. The fact that maximizing V in the *short term* is equivalent to maximizing R in the *long term* makes dynamic programming particularly attractive. Dynamic programming based on the optimality equation (eq. 3.1-4) solves, at least in principle, a variety of important problems. The following sections describe several well-known computational methods of stochastic dynamic programming.

3.2 Synchronous Value Iteration

This section describes a successive approximation approach for obtaining the optimal value function—*synchronous value iteration* (SVI). It is a successive approximation method for solving the optimality equation (eq. 3.1-4).

In [Ross83], it is shown that when the decision process is in state *x*, a policy π that chooses the action that maximizes the right-hand side of the optimality equation (eq. 3.1-4)

$$R(x,\pi(x)) + \gamma \sum_{y} P_{xy}(\pi(x)) V^{*} = \max_{a} \left\{ R(x,a) + \gamma \sum_{y} P_{xy}(a) V^{*}(y) \right\},$$

is an optimal policy. It follows immediately that we would know the optimal policy should we know the optimal value function V^{*}. The value iteration method for obtaining the optimal value function V^{*} is as follows: let $V_0(x)$ be any arbitrary bounded function, and define the recursive functional

$$V_n(x) = \max_{a} \left\{ R(x,a) + \gamma \sum_{y} P_{xy}(a) V_{n-1}(y) \right\} \qquad \text{(eq. 3.2-1)}$$

for $n > 1$. It can be shown [Ross83] that $V_n(x) \to V^{*}(x)$ uniformly in x as $n \to \infty$, assuming $R(x,a) < N$ for all x and a. The method is said to be synchronous because the computation of V_n depends only on the values of V_{n-1}.

Thus, to find an optimal policy, we first approximate the optimal value function, V^{*}, by value iteration, *i.e.*, the repeated applications of (eq. 3.2-1), and then determine an optimal policy to be the one that selects the action that maximizes the right-hand side of the optimality equation (eq. 3.1-4) for each state.

It can be shown that although V_n is not necessarily closer to V^{*} than V_{n-1} over all states, the maximum error between V_n and V^{*} must decrease. Mathematically, the SVI is a contraction mapping with respect to the L_{∞} norm and has V^{*} as its unique fixed point.

3.3 Policy Improvement

In addition to the type of approximation for constructing an optimal policy discussed above, *policy improvement* or *policy iteration* is a method of successive approximations in policy space for producing optimal policies [Bellman61]. It is based on the observation that one only needs to know the value function for a given policy in order to improve that policy.

Suppose that policy π_g is some proposed policy, and we wish to know if it is better than the existing policy π_f, for which V^{π_f} is known, *i.e.*, will π_g yield higher expected returns for all states than π_f? One way to decide if π_g is better than π_f would be to compute V^{π_g} by solving the system of linear equations defined by

$$V^{\pi}(x) = R(x,\pi(x)) + \gamma \sum_{y} P_{xy}(\pi(x)) V^{\pi(y)} \qquad \text{(eq. 3.3-1)}$$

and then compare V^{π_g} with V^{π_f} over all states. However, computing the value function is computationally costly. A more efficient way of doing this and one that does not require computing V^{π_g} is as follows. Consider the expected return, starting in state x, that would result from following policy π_g for one step, and then switching to π_f thereafter:

$$V^{\pi_{gf}}(x) = R(x,\pi_g(x)) + \gamma \sum_{y} P_{xy}(\pi_g(x)) V^{\pi_f}(y) \qquad \text{(eq. 3.3-2)}$$

If $V^{\pi_{gf}}(x) \le V^{\pi_f}(x)$ for all states x with strict inequality for at least one state, then it can be shown that π_g is indeed an improvement over π_f.

The foregoing ideas provide the basis for the policy improvement method. Instead of using (eq. 3.3-2) to find out if a proposed policy is an improvement over the current one, the policy improvement method uses it to define a new policy that is an improvement over the existing policy. The new policy is defined as the one that for each state x, chooses the action, $\pi_g(x)$, that maximizes the right-hand side of (eq. 3.3-2). It can be shown [Ross83] that $V^{\pi_g}(x) \le V^{\pi_f}(x)$ for all states, and if $V^{\pi_g}(x) = V^{\pi_f}(x)$ for all states, then $V_{\pi_g} = V_{\pi_f} = V^*$. The policy improvement method is summarized in Figure 3.3-1.

- Let π_0 be an arbitrary initial policy.
- $i \leftarrow 0$.
- Repeat
    1. $i \leftarrow i + 1$.
    2. Compute the value function, $V^{\pi_{i-1}}$, for π_{i-1} by solving the system of linear equations defined by (eq. 3.3-2).
    3. Let π_i be the policy that, for each state x, chooses an action, a, that maximizes $R(x,a) + \gamma \sum_{y} P_{xy}(a) V^{\pi_{i-1}}(y)$.
- Until both π_i and π_{i-1} are optimal.

Figure 3.3-1. Policy Improvement Algorithm

An informal argument for the result goes like this: starting at state x, it is better to follow π_g for one stage and then follow π_f thereafter than to follow π_f throughout. But by the same token, it is better to follow π_g for one further stage from the state just reached. Repeating this argument shows that it is always better to follow π_g than it is to follow π_f.

3.4 Asynchronous Value Iteration and Q-Learning

The classic method of value iteration described in section 3.2 requires that V_n is computed for all states using the values of V_{n-1}, and then V_{n+1} is computed using the values of V_n, and so on. However, the value iteration method need not be carried out in this way. In fact, the values of individual states can be updated in an arbitrary order. This manner of computing V^* is called *asynchronous value iteration* (AVI) [Bertsekas87,BertsekasTsitsiklis89]. AVI is appealing since the convergence result requires only that the values of all states be updated often enough [BertsekasTsitsiklis89,Watkins89], and thus leaves a variety of ways for the use of heuristics to control the process. In practice, this simply implies that whatever strategy is used to choose states whose values are to be updated, no state should suffer from starvation, i.e., no state should ever be prevented from being chosen in the future.

It is important to realize that each individual update of a state's value in AVI does not necessarily make it more accurate as an estimate of the state's true value. However, the estimated value function will converge to the true value function as the total number of updates tends to infinity. It should also be noted that although the order in which the values of states are updated does not alter the convergence result of AVI, it does influence the rate of convergence.

This form of incremental DP provides a basis on which a variety of learning and planning systems can be developed. For example, a popular TD method, Q-learning [Watkins89], can be viewed as incremental, Monte-Carlo value iteration.

Finally, there are other versions of the value iteration method that are somewhere between the SVI and the total AVI, such as Gauss-Seidel value iteration. In the Gauss-Seidel method, the values are updated one state at a time, sweeping through all the states in a sequential manner, with the computation for each state using the up-to-date values of the other states. Like AVI, the order in which the values of states are updated influences the computation. Gauss-Seidel value iteration converges to the true value function under the same conditions under which SVI does.

3.5 Discussion

So far we have only discussed the situation where state or action spaces are discrete. When the state or action spaces are continuous, dynamic programming is typically applied by first quantizing the state or action spaces and then applying standard algorithms such as the value iteration method to the quantized state or action spaces as if they were discrete problems.

It is important to realize that although dynamic programming is much more computationally tractable than explicit exhaustive search of all possible state sequences, the nature of the exact computation of dynamic programming is still exhaustive in that it requires the explicit enumeration of all possible states. However, exhaustive computation is very impractical for problems with large number of states, which are typical in real world applications. For this reason dynamic programming has not played a significant role in artificial intelligence research.

In terms of dimensionality of the state space, the practical limitation of dynamic programming can be quantitatively described as follows. As the number of states increases exponentially with increasing dimensionality, so does the complexity of time and space for dynamic programming to produce an optimal solution, and usually reaches a practical limit when the dimension of the state space is approximately five or six [Fu70]. Approaches that enumerate all possible actions increase similarly in complexity with the dimension of the action space. This problem is also referred to as Bellman's *curse of dimensionality*. It was first discussed by Bellman [Bellman61] and has remained the central issue in dynamic programming since.

In practice, this simply means that the state space may be too large to allocate memory to the entire state space or to update all states repeatedly as required by the value iteration method. Unless the dimensionality problem can be adequately addressed, the practical utility of dynamic programming remains limited. One of the aims of this proposed effort is to look for ways that make dynamic programming more practical and commercially feasible.

Section 4: REINFORCEMENT LEARNING FOR GAZE CONTROL

Foveal vision simultaneously supports a wide FOV, localized high acuity, and high temporal resolution while minimizing sensor data to that which is relevant [Bandera92]. Additionally, its scale space representation of objects is highly effective for real-time ATR applications. However, space-variant data acquisition necessitates the development of efficient gaze control mechanisms and rapid schemes for referencing object identities to improve real-time performance and fidelity. This section presents a commercially feasible reinforcement learning method for gaze control, together with empirical studies using an active visual sensor in a simulated foveal ATR problem, with application to autonomous mobile platforms.

4.1 Problem Description

One of the main attributes of the ATR problem we study is that multiple fixation points, and the accumulation/integration of relevant visual information acquired from each gaze, will be required to classify a target (to within some confidence threshold). This requires the scale-space volume defined by the discriminant target features to be greater than the scale-space volume supported by the foveal retinotopology [Bandera94]. Figure 4.1-1 shows the simulated ATR problem used for the experiments reported here.

Each graph in Figure 4.1-1 represents an ROI where a potential target has been preattentively located. Each target is represented by four features (1 or -1) placed in various locations in the ROI. Each feature carries equal importance in determination of a target. It should be noted that both classification features and their locations (highlighted boxes) are important for determining a target. However, the precise feature position in each highlighted box is unimportant. This allows the classification module to tolerate possible sensor-reading errors in actual ATR tasks. The foveal agent has limited sensing capabilities. The objective is for the agent to interrogate and extract spatial patterns in a scale space fashion with a minimum amount of computation, thereby forming high dimensional response vectors that are then used to rapidly form an association with the object identities.

					-1
	-1				
				1	
		1			

		1			
	1				
-1					-1

				-1	
		1			1
-1					

		1			
1					
		1			-1

Figure 4.1-1. Simulated ATR Problem

The foveal agent is equipped with an active visual sensor with a local receptive field, shown in Figure 4.1-2, that it can manipulate using a given set of four control actions. These actions can steer the optical head of the sensor in four compass directions: *left*, *right*, *up*, and *down*. The active sensor has two type of sensing devices, both of which have a local receptive field of the highlighted boxes. The center one, however, has a higher resolution capable of encoding the type of a feature and its precise location. The peripheral sensing device has a low resolution. It can only encode feature types.

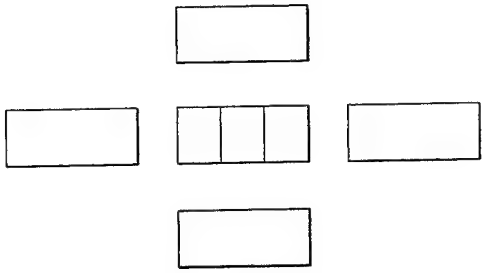


Figure 4.1-2. Active Foveal Sensor with a Local Receptive Field

One can pose gaze control (saccades) for the ATR task as a learning problem in many different ways. In this report gaze control for active vision is formulated as a reinforcement learning problem. The short-term reward for each action is zero except the one that leads to a target recognition. An alternative reward structure is on-line reinforcement, which accelerates convergence of the value function [Bandera96]. The performance measure is total discounted reward received over time. Thus, the optimal performance according to this metric is for the agent to choose the shortest gaze (saccades) sequence required for a target recognition.

It is important to note that since the agent has only limited sensing capabilities in terms of the local receptive field, the problem is no longer Markov in nature. That is, the agent is unable to distinguish between similar world states. The problem is further compounded by the fact that visual information acquired from each fixation may not be sufficient to decide where to look next, since different targets may share similar features at similar locations. It is a kind of active perception task in which the agent has to manage its sensing operations in order to accumulate adequate state information for making decisions. Therefore, the agent must construct its internal states summarizing what it has seen to date (historical information) to cope with the perceptual aliasing.

4.2 Algorithms Used

On large problems, reinforcement learning systems must use compact approximation schemes to represent value functions. There are several approximation schemes, such as CMACs, RBFs, and neural networks, that are capable of real-time performance, which is essential in many time-critical applications. The particular function approximator we use is neural networks. Neural networks are general approximators and are capable of implementing any function to any desired degree of accuracy given sufficient resources. In addition, there exist learning techniques well-suited for recurrent networks that are required for many tasks involving time, including the gaze control problem we study here.

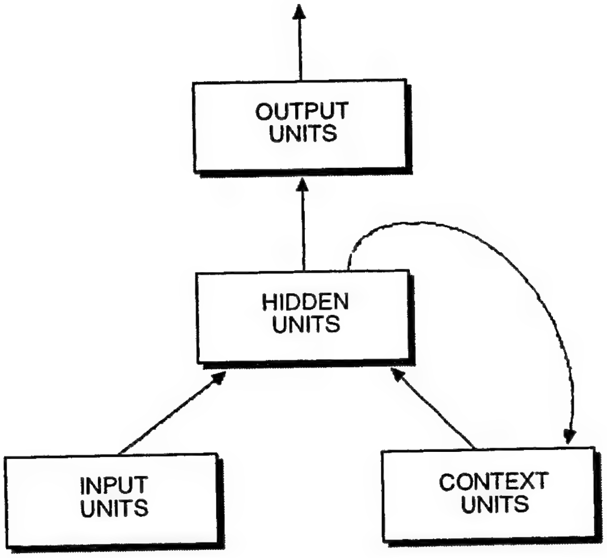


Figure 2.2-1. An Elman Net

Recurrent neural networks, such as Elman networks, provide a way to construct useful history features that are essential for solving partially observable Markov decision problems. As shown in Figure 4.2-1, input to an Elman network consists of two parts: input units and context units.

The context units carry feedback activations resulted from the network state at the previous time step. That is, the context units at time step *t* are copies of the hidden units at time step *t - 1*. As a memory structure, the context units remember an aggregate of previous network states. As a result, the behavior of the network depends on past as well as current inputs.

Adjustable weights in an Elman network are those between the input units and the hidden units, those between the context units and the hidden units, and finally those between the hidden units and the output units. To predict utility values correctly, the recurrent network will be forced to discover historical features that permit the network to properly assign utility values to inputs displaying the same perception.

The recurrent network is trained using the SARSA algorithm in conjunction with back-propagation. SARSA is a $Q(\lambda)$ based multi-step learning technique [PengWilliams94,96, RummryNarajan94]. The SARSA method has shown dramatic performance improvement over Q-learning on all the tasks that have been examined so far [PengWilliam94, RummryNarajan94, Sutton95]. In Q-learning, error correction is made only to the Q value estimate of current state-action pair. To make correction to earlier actions, many trials will have to take place. In contrast, Q(λ) based techniques use a trace mechanism, which is a memory structure, in such a way that error correction can be made not only to the Q value estimate of current state-action pair, but also to that of earlier state-action pairs. This in a sense amounts to efficient use of experience and parallels only to that of EBL [Mitchell86]. More recently, Sutton [Sutton96] argues that in an environment where learning takes place continuously, as it should, SARSA has the advantage of following the policy it is actually estimating, whereas Q-learning estimates a policy that it does not follow. In terms of average reward received per trial, SARSA performs significantly better than Q-learning. The SARSA algorithm is described in Figure 4.2-2, where *W* is the weight vector, *Q*, parameterized by *W*, is the current approximation to action values, *r* is the immediate reward, γ is the discounted factor, α is the learning rate, and λ is a procedure (meta) parameter that controls credit distribution over time.

```
1. Reset all eligibilites, $e_0 = 0$
2. $t = 0$
3. Select action, $a_t$.
4. If $t > 0$ then $W_t = W_{t-1} + \alpha\left(r_{t-q} + \gamma Q_t - Q_{t-1}\right)e_{t-1}$
5. Calculate $\nabla_w Q_t$ w.r.t. selected action $a_t$ only.
6. $e_t = \nabla_w Q_t + \gamma\lambda e_{t-1}$
7. Perform action $a_t$, and receive short-term reward $r_t$.
8. If trial has not ended, $t \leftarrow t+1$ and go to step 3.
```

Figure 4.2-2. The SARSA Algorithm

4.3 Simulation Results

This section reports the experimental results of the SARSA algorithm combined with recurrent networks for learning optimal gaze control in the foveal ATR problem described in section 3.1. Through these experiments, we expect to gain more insights into how well these learning systems work in different conditions.

After some experimentation, we decided on an Elman network with 10 input units, six hidden units, six context units, and one output unit that encodes action values for a given state-action pair. Note that the number of hidden units could have been determined via cross-validation. The input units are as follows:

- three units: These are the inputs to the center sensing device. One encodes information about the feature being detected. Two units encode the location of the feature.
- four units: Each of these units represents a feature being detected by each of the peripheral sensors.
- two units: These encode four control actions.
- one unit: The bias unit is always on.

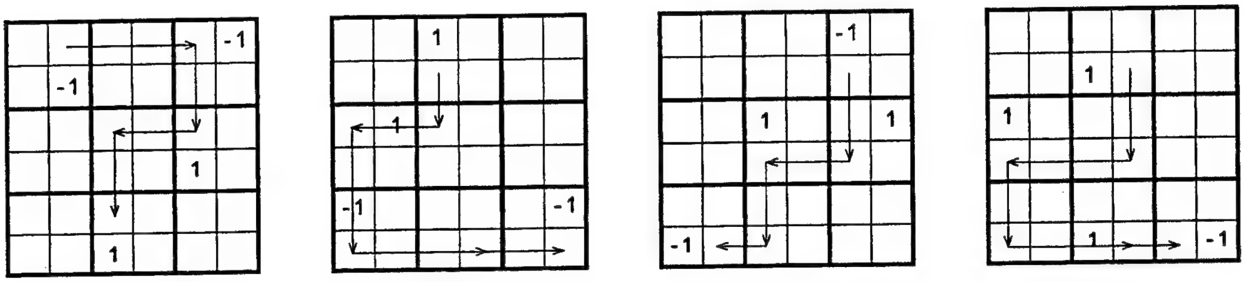
The experiment consisted of a series of trials. A trial here is a sequence of gazing fixations. A fixation sequence begins with the active sensor pointing at a starting location within the ROI. It ends when a target has been recognized. In the ATR problem described above, recognition is successful when all four features have been detected and classified.

The weights are initialized to lie between -0.1 and 0.1 and are updated after each gaze. This is in direct correlation to the typical weight update procedure in neural networks in which weights are updated according to the stochastic gradient or incremental procedure instead of the total gradient rule [LeCun91]. That is, updates take place after each presentation of a single exampler without averaging over the whole training set. Both empirical and theoretical studies show that the stochastic gradient rule converges significantly faster than the total gradient rule, especially when training set contains redundant information.

There are several procedural (meta) parameters that have to be determined experimentally. These are the following: the discounted factor γ, the weighting parameter λ, and the learning rate α. Values for these parameters were 0.9, 0.3, and 0.5. The experimental tests showed that good performance can be obtained under a wide range of these parameter values. In addition, during training, the agent employs an ε-greedy policy to ensure sufficient exploration. An ε-greedy policy is one that, at each state, chooses greedy actions (1 - ε) percent of the time and random actions ε percent of the time. In the experiments reported within this report, ε was chosen to be 0.15 throughout.

Figure 4.3-1 shows optimal gaze sequences for each ROI by the control policy after 15,000 trials. These gaze sequences give rise to successful target recognition for each task. Note that optimal gaze sequences are not unique. Note also that for each recognition task the agent starts saccading at an arbitrarily chosen feature location. This initial pre-attentive behavior is well-founded since the very fact that the ROI was identified indicates the presence of features.

An additional experiment, whose detail we omit here, was carried out in which some feature locations in highlighted boxes were displaced. The intent here was to model noise sensor reading and possible distortion in a perception process under real-world conditions, and to evaluate the robustness of the learned gaze control policy. The result showed that the same gaze sequences were indeed recommended by the control policy. This indicates the generalizing capability of the learning system.



FOVEAL-378-100396

Figure 4.3-1. Fixation sequences recommended by the optimal gaze control policy

To further illustrate the characteristics of the learning system, the learning curve for the system was graphed in Figure 4.3-2. As a comparison, the performance of Q-learning as a function of time was also graphed. The learning curves were an average over 10 runs. Both algorithms used the same sets of initial random weights. Figure 4.3-2 shows clearly the superior performance of SARSA over Q-learning. Furthermore, SARSA exhibits more stable behavior than Q-learning despite the fact that both employed the same ε-greedy policy. The result is consistent with prior empirical studies by others. In addition, to the best of the author's knowledge, this is the first documented empirical study that combines SARSA with recurrent neural networks.

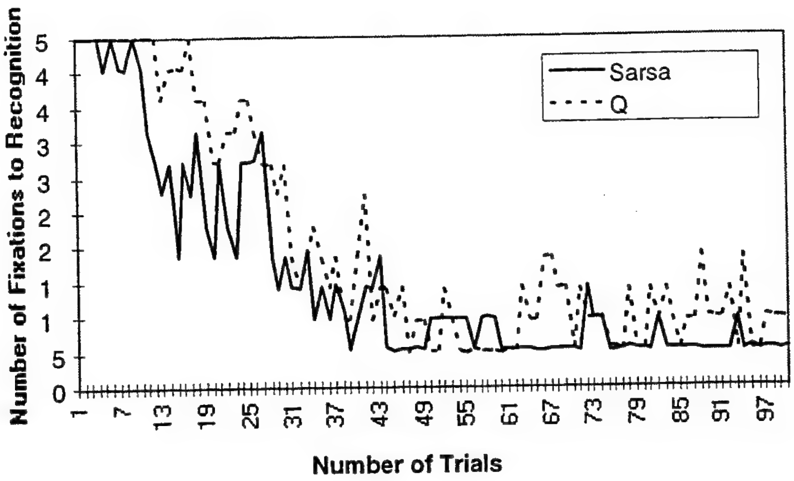


Figure 4.3-2. Performance of gaze control policies by SARSA and Q-learning

4.4 Discussions

We have presented a reinforcement learning method for gaze control for active vision and empirical evidence supporting the claim that reinforcement learning is a commercially feasible learning control technology for use in foveal ATR.

We showed that reinforcement learning in conjunction with recurrent neural networks is capable of solving active perception tasks where perceptual aliasing is prevalent and poses significant challenges to traditional reinforcement learning methods. Our empirical results treated a much more realistic case requiring multiple fixation points to classify a target. These results showed that the optimal gaze control strategy in a partially observable Markov environment can be successfully learned using reinforcement learning. Our empirical results also showed that SARSA offers significant improvements in learning speed and reliability over Q-learning.

Reinforcement learning for gaze control provides large advantages over non-learning control methods in terms of adaptability, efficiency, reliability, and commercial feasibility.

Section 5: UB'S CONTRIBUTION

UB's effort in this program is focused on target detection in active machine vision, which is a fundamental building block for foveal multiple target tracking and/or ATR.

The nature and complexity of target detection using a visual sensor makes it very difficult, if not impossible, for humans to construct an optimal visual sensor control routine (policy) for robots to detect the target efficiently. By applying reinforcement learning to this problem, the detection system can be expected to produce a satisfactory performance with respect to the sensor's physical parameters. The basic reinforcement learning algorithm, namely Q-learning, was evaluated in this context and applied to a simulated detection problem. It is important to note that this target detection problem is inherently non-Markov due to robot limited sensing capabilities. UB proposed several memory-based approaches, such as *direction-register* and *memory-map,* to the problem. These techniques improve upon simple Q-learning and demonstrate the following:

1. Reinforcement learning is able to produce a policy that has a much improved performance for solving the target detection task.

2. Direction-register and memory-map are promising methods for solving partially observable Markov decision problems that are typical of real-world applications.

3. The hierarchical control strategy (e.g., separate low-level controllers for different visual acuities and an upper level gating routine to decide the priority) seems very promising for solving multi-resolution visual attention, such as those involving foveal sensors.

UB also investigated input generalization and attempted several methods such as learning vector quantization (LVQ) and the adaptive Resonance Theory (ART) architecture to address the issue. For further details, see attached UB report.

Section 6: EXPERIMENTS WITH RESIDUAL AND SARSA LEARNING

The previous sections demonstrate the feasibility and efficacy of reinforcement learning coupled with recurrent neural networks as a gaze control mechanism for active perception. This section investigates several computational issues of reinforcement learning with function approximation within the context of behavior learning and selection. The objective is to examine the relative learning efficiency of these systems, which bears significant importance to time-critical applications both commercial and military.

6.1 Reinforcement Learning with Function Approximation

Most reinforcement learning methods for autonomous systems require the identification of each world state, and look-up table representations for policies and value functions, in order for them to converge to optimality. Except for a very limited number of applications, however, look-up table representations are fundamentally unsuitable for learning in most practical problems. First, the state space of real world problems is often too large to allocate memory to the entire state space. As the dimension of the state space increases, the storage requirement becomes intractable. Second, look-up table representations typically learn the values of states one at a time, and the value of one particular state does not automatically influence the values of similar states. However, the state space of large-scale problems prohibits a learning system to visit all of the states, let alone update each one repeatedly to determine its correct value. Thus, in general, lookup tables do not scale well for high dimensional problems (the *curse of dimensionality*).

There is a variety of methods that can represent value functions more efficiently and that are capable of valid generalization [Barnhill77,Franke82,Schumaker76]. One extreme case is when a closed-form expression can be found for the value function. However, the problems in which this can be performed are quite limited, such as when the world model is linear and the performance criterion is quadratic. Other methods settle for trying to represent the value function approximately. The basic principle of these approximate methods is to trade off optimality for efficiency, much like heuristic search algorithms do in artificial intelligence research. One approach of this type is to use a multigrid version of successive approximation (value iteration) that proceeds from coarse to fine grids. Another approach is to assume some underlying functional form for the value function and try to compute which actual function of this form best fits the Bellman equation as applied to actual data.

The fundamental difficulty with reinforcement learning with compact function approximation is that convergence to optimality, as in the case of lookup tables, is no longer guaranteed. When combined with reinforcement learning, some such methods have been shown to be unstable in theory [Baird95,Gordon95] and in practice [BoyanMoore95]. On the other hand, other methods have been proven stable in theory [Sutton88,Peng94] and very effective in practice [Lin91,Tesauro92]. What are the requirements of a method in order to obtain good performance? In the following sections, we analyze two principal reinforcement learning algorithms, namely: the residual gradient algorithm and the SARSA algorithm, both of which can be shown stable in theory [TsitsiklisRoy96] and in practice [Sutton95].

6.2 Residual Algorithms and SARSA

For a prediction problem with a finite number of states, the optimal value function is the function that uniquely satisfies the Bellman equation:

$$V(x) = \langle r + \gamma V(y) \rangle \qquad \text{(eq. 6.2-1)}$$

where $\langle\ \rangle$ denotes expectation and y is a successor state for a given state x. The *Bellman residual* (or *error*) for a value function V is defined to be the difference between the two sides of the Bellman equation (eq. 6.2-1). The *mean squared Bellman error* for a prediction problem with n states is thus:

$$E = \frac{1}{n}\sum_{x}\left[\langle r + \gamma V(y) \rangle - V(x)\right]^2 \qquad \text{(eq. 6.2-2)}$$

The residual gradient algorithm [Baird95] attempts to minimize the Bellman error by performing gradient descent on the mean squared Bellman error, E (eq. 6.2-2).

For a deterministic prediction problem, weight change ΔW, after a transition from state x to state y, will be made according to:

$$\Delta W = -\alpha\left(r + \gamma V(y) - V(x)\right)\left(\nabla_w \gamma V(y) - \nabla_w V(x)\right) \qquad \text{(eq. 6.2-3)}$$

where α is the learning rate. It is clear that a prediction problem with a finite number of states performing gradient descent on E guarantees convergence to a local minimum.

For a control problem with a finite number of states, the mean squared Bellman error will be as follows:

$$E = \frac{1}{n}\sum_{x}\left[\langle r + \gamma V(y)\rangle - Q(x,a)\right] \qquad \text{(eq. 6.2-4)}$$

where $V(x) = \max_a Q(x,a)$. Similar to the deterministic prediction problem, weight change ΔW in a deterministic control problem, after a transition from state x to state y resulting from taking action a, will be made according to:

$$\Delta W = -\alpha\left(r + \gamma V(y) - Q(x,a)\right)\left(\nabla_w \gamma V(y) - \nabla_w Q(x,a)\right) \qquad \text{(eq. 6.2-5)}$$

This will give rise to a similar algorithm performing gradient descent on E (eq. 6.2-4).

It is important to note that the derivations of (eq. 6.2-3) and (eq 6.2-5) assume that the dynamics of the underlying system is deterministic. If the dynamics were stochastic, the residual gradient algorithm would still converge, but not to a local minimum of the mean squared Bellman error. It will be something else, as was noted by Werbos [Werbos90]. Despite this, it has been suggested [Baird95] that the sheer convergence itself might still be a strong argument in favor of the algorithm as a viable candidate for solving stochastic prediction or control problems. We were curious to test this hypothesis.

For a stochastic problem, update rules (eq. 6.2-3) and (eq. 6.2-5) in general will have to be modified according to:

$$\Delta W = -\alpha\left(r + \gamma V(y) - V(x)\right)\left(\nabla_w \gamma V(y') - \nabla_w V(x)\right)$$

and

$$\Delta W = -\alpha\left(r + \gamma V(y) - Q(x,a)\right)\left(\nabla_w \gamma V(y') - \nabla_w Q(x,a)\right)$$

where y and y' are two successor states of x, each drawn independently from the distribution defined by the stochastic problem. This is required because an unbiased estimator of the product of two random variables can only be obtained by multiplying two independently generated unbiased estimators. To accomplish this a model of the problem must be known, either *a priori* or learned. This clearly increases the computational complexity of the algorithm. In addition, acquiring a model of the world may prove difficult, if not impossible, for many practical applications where state space is continuous. Furthermore, given that model learning is tractable, the algorithm might still be affected in undesired ways by errors in model acquisition, as shall be seen in the following sections.

The SARSA algorithm we use here is described in Figure 4.2-2.

6.3 The Random Walk Problem

The random walk problem, shown in Figure 6.3-1, is a simple prediction problem. There are seven states in which state D is the starting state. At each state a transition is made either to the left or to the right with equal probability except states A and G, where the state transition is always to itself, i.e., states A and G are absorbing. The short-term reward is zero for every transition except the one from state F to state G when the reward 1 is delivered. The objective is to predict at each state the probability of reaching state G.

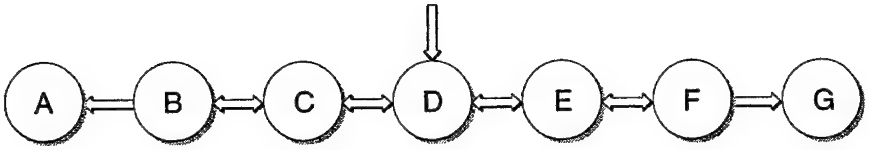


Figure 6.3-1. A random walk problem

Previous studies [Sutton88,Sutton96] of the random walk problem have used lookup tables to represent the value function. In this experiment, the value function is represented by

$$V(x) = wx$$

where w is a free parameter or weight, and x represents state. That is, V is a linear function of its input. States are encoded by integers from 0 to 7 corresponding to states from A to G, assuming values of states A and G are always zero. Note that the learning agent has no direct knowledge of what state it is in at each given moment. Instead, it only sees the encoding of the states. A trial begins at state D and ends when either state A or state G is reached. The weight change is made after each transition.

For the random walk problem, the value of each state can be computed exactly. The ideal predictions for each of these states from B to F are: 1/6, 1/3, ½, 2/3, and 5/6. For a given approximation V to the ideal predictions, root mean-squared-error (RMSE) can be computed as a performance measure. The RMSE is computed anew after every 20 backups instead of a trial, where a backup is defined here to be a single weight update. The main reason is objectiveness. A close look at the residual gradient algorithm reveals that the update rule (eq. 6.2-3) can be rewritten as:

$$\Delta W = -\alpha\big(r + \gamma V(y) - V(x)\big)\nabla_w \gamma V(y) + \alpha\big(r + \gamma V(y) - V(x)\big)\nabla_w(x)$$

which amounts to two backups in an update equation, as opposed one in SARSA learning. On the other hand, since there is only one weight in the approximation system, trace computation is negligible.

Figure 6.3-2 shows the performance of the random walk problem by SARSA, residual gradient with a learned model, and residual gradient without a model. The procedural (meta) parameters for these algorithms are: $\lambda = 0.55$, $\alpha = 0.002$ (SARSA); $\alpha = 0.016$ (residual with a model); $\alpha = 0.02$ (residual without a model). These curves are averaged over 50 runs. All the algorithms used the same initial random seeds. Figure 6.3-2 illustrates clearly the superior performance of the SARSA algorithm over the residual gradient algorithm. Both the SARSA and residual algorithms, with a learned model, continued to improve as learning proceeds. On the other hand, the residual gradient algorithm without a model failed to improve even when the system was let run sufficiently long. This result suggests strongly that for a stochastic problem a model is critical, be it learned or *a priori*, for the residual gradient algorithm to converge to a minimum of the mean squared Bellman error.

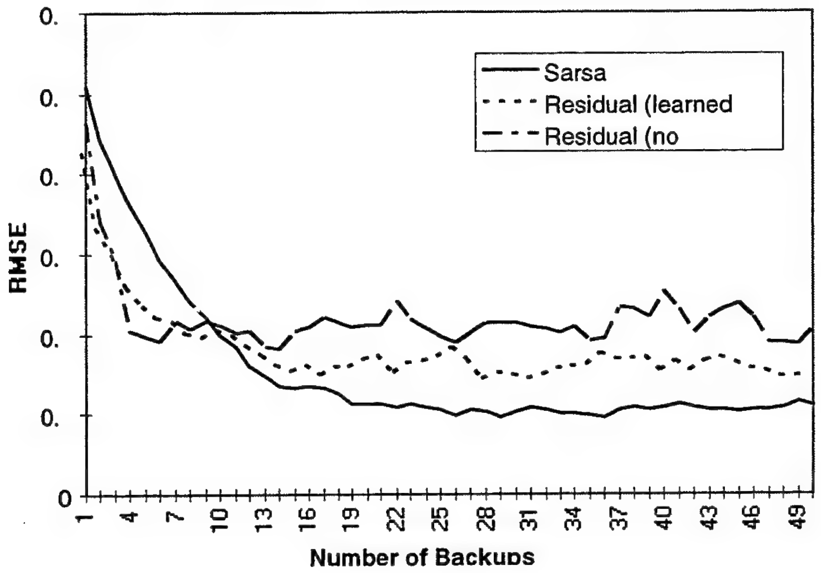


Figure 6.3-2. Performance of random-walk by SARSA, Residual with a learned model, and Residual without a model

6.4 The Two Dimensional Stochastic Continuous Grid-World Problem

To further evaluate the relative efficiency of these algorithms, an experiment was carried out in a two dimensional stochastic continuous grid-world domain. The two dimensional continuous grid-world is shown in Figure 6.4-1. In the grid-world, the state is a point $(x, y) \in [0,2]^2$. There are four actions corresponding to short steps (length 0.1) in each compass direction. To complicate the matter, however, each action has the probability of only 2/3 to actually move the agent in an intended direction, and the probability of 1/3 in either perpendicular directions. For example, suppose the agent takes an action having an eastward motion. The probability that the agent actually moves one step to the east is 2/3. The agent may very well land at a state one step north or south from its current state with the probability of 1/6.

The start region is $[0,0.1]\times[0.2,0.3]$ and the goal region is $[1.8,2]\times[1.0,1.2]$. There is a thin barrier $0.3\times[0.1,1.6]$ that cannot be crossed by the agent. Any action that would ostensibly move the agent into the barrier or outside the grid-world boundary has the actual effect of keeping the agent at its current location. The short-term reward is 0 for each action except for those that lead to the goal region when a reward of 10 is delivered.

The agent initially has no knowledge of the effect of its actions on what state it will occupy next, although it always knows its current state. Loosely stated, the objective is for the agent to discover a policy that enables it to obtain the maximum rate of reward received over time. More specifically, every time the agent occupies the goal region we may place it back in the starting region. In this case, the agent's objective is simply to discover the shortest path from the starting region to the goal region.

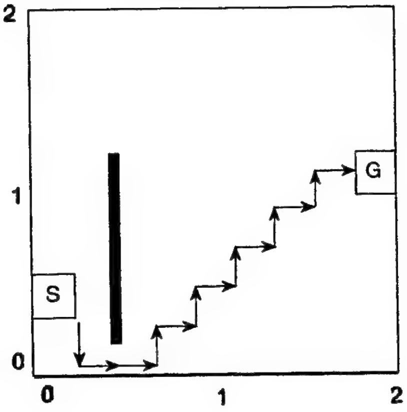


Figure 6.4-1. A two dimensional stochastic continuous grid-world

All of these methods apply more generally to a much wider range of learning control tasks, in which case the two dimensional continuous grid-world is simply a conceptual stand-in for the appropriate abstract state space for the actual problem and the compass direction moves used here represent the various control actions available.

The grid-world task has a continuous two dimensional state space with an infinite number of states. To apply reinforcement learning some form of a function approximator is required. We used a set of four CMACs, one for each action. Unlike neural networks, which are global function approximators in that each hidden unit has a global receptive field, CMACs are sparse coarse coded function approximators. In particular, CMACs use a hashing algorithm to implement a distributed look-up table representation and a fixed amount of overlap of look-up table weights to reduce table size and generalization.

A CMAC uses multiple overlapping tilings of the state space to produce a feature representation for a final linear mapping where all the learning takes place. See Figure 6.4-2. The overall effect is much like a RBF network with fixed radial basis functions, except that it is particularly efficient computationally. The significance of CMACs can be seen in recent work

[TsitsiklisRoy96] by Tsitsiklis and Van Roy. It is shown that for a fixed set of tilings (basis functions) from the CMACs, when coupled with TD(λ) returns, are guaranteed to converge to optimality in the LMS sense. Although this result applies only to prediction problems, it does shed light on control problems, as well. It shows us that for any approximation scheme, convergence cannot be expected when updates do not follow actual sample trajectories.

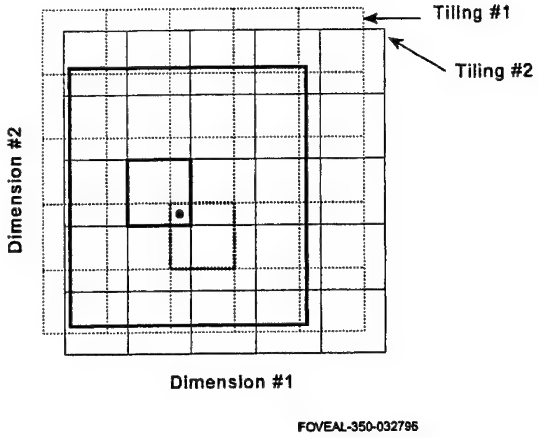


Figure 6.4-2. CMACs involve multiple overlapping tilings of the state space

In the grid-world problem we divided the two state variables each into ten evenly spaced intervals, thereby partitioning the state space into 100 regions or boxes. We added an eleventh row and column so that the titling could be offset by a random fraction of an interval without leaving any state uncovered. This process was repeated five times, each with a different, randomly selected offset. The result was a total of 11 × 11 × 5 = 605 boxes. The state at any moment was represented by the five boxes, one per tiling, within which the state resided. One can think of the state representation as a feature vector with 605 features, exactly five of which are present at any point in time. The approximate action values are linear in this feature representation. Note that this representation of the states results in the violation of the Markov property: many different nearby states produce exactly the same feature presentation.

It is important to note that the tilings need not be grids. For example, to dodge the "curse of dimensionality," a possible technique is to ignore some dimensions in some tilings, i.e., to use hyperplanar slices instead of boxes. A second major technique is "hashing," or varying-resolution, - a consistent random collapsing of a large set of tiles into a much smaller set. Through hashing, memory requirements are often reduced by large factors with little loss of performance. This is possible because high resolution is needed in only a small fraction of the state space, i.e., those where value functions are not smooth. Hashing liberates us from the curse of dimensionality in the sense that memory requirements need not grow exponentially with dimensionality, but need merely meet the real demands of the task.

An ε-greedy policy was used to choose actions, where ε was set to 0.15 in all the experiments. We used the heuristic to ensure sufficient exploration. The initial weights were set to lie between -0.1 and 0.1. A trial begins with the agent at a state in the starting region and ends when a state in the goal region has been reached. A starting state was uniformly, randomly generated from the starting region.

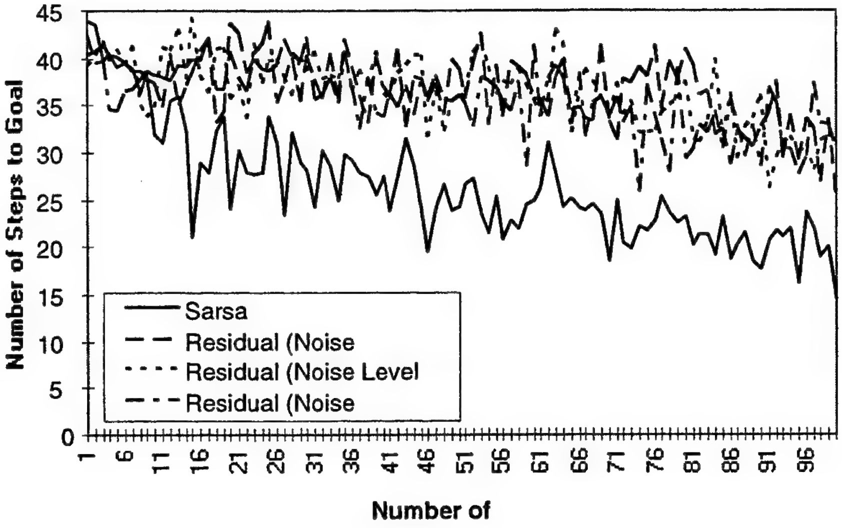


Figure 6.4-3. Performance of the continuous grid-world problem by SARSA and residual gradient algorithms

We applied SARSA and residual gradient algorithms to this task, each with procedural parameters selected after extensive search to give the best performance of each algorithm. Each algorithm was run for 100 trials. All algorithms used the same sets of random starting states and the same sets of initial random weights. The performance measure was the number of steps it takes for the agent to reach a goal state. Because of the stochastic transition nature, these steps were averaged over 10 traversals. That is, after each trial, the agent was placed at a starting state and then the greedy policy was followed until a goal state was reached or some maximum number of steps (500) was exceeded. This was done 30 times, and the average path length was recorded. This measure was then averaged over 10 runs to produce the results shown in Figure 6.4-3.

Several results are evident from Figure 6.4-3. First, the SARSA algorithm performed significantly better than the residual gradient algorithms. In particular, the results produced by the residual gradient algorithms assumed a known model available. If the residual gradient algorithms had to learn the model on-line, the results could have been worse. In addition, model learning would increase the overall computational complexity of these algorithms. To see how a model might affect performance, errors were added to the model. In general, errors may occur in two places: one in state transition estimation and the other in transition probability estimation.

These errors always exist, particularly in high dimensional or continuous state spaces where compact approximation schemes must be used.

The performance of the residual gradient algorithms under erroneous model conditions is also shown in Figure 6.4-3. It can be seen that the performance degrades with increasing noise levels. Furthermore, the performance showed little or very slow improvement as more trials took place. This further verifies the results presented in the previous subsection. It casts serious doubt on applying residual gradient algorithms to solve stochastic Markov decision problems. In contrast, a model-free method such as SARSA would stand a much better chance to succeed in these problem domains.

Figure 6.4-4 plots the value function computed by SARSA after 100 trials. Similarly, Figure 6.4-5 plots the value function computed by the residual gradient algorithm using perfect model information. Clearly, SARSA created a better surface than the residual gradient method. The large valley area in the middle of the state space created by the residual gradient algorithm can cause the agent to move in a wrong direction, which certainly contributes to the relatively poor performance. In addition, the same value function shows flatness in the vicinity of the starting region, which is in sharp contrast with the value function computed by SARSA. Therefore, a "residual" agent can take a step in either direction instead of the preferred southward direction.

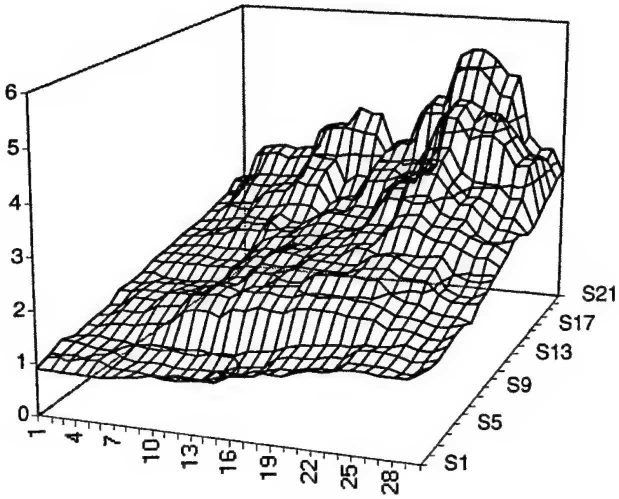


Figure 6.4-4. The value function computed by SARSA after 100 trials

Much more empirical work is needed with these algorithms before a definitive conclusion can be drawn about their relative efficiency, particularly when function approximators are used. However, these experiments do provide strong evidence for two points: (1) the value function produced by SARSA is a better approximation to the optimal value function than that generated by the residual gradient algorithm, all other things being equal, and (2) the residual gradient algorithms require a model in solving stochastic Markov decision problems, which causes these methods to be susceptible to unavoidable errors in model estimation.

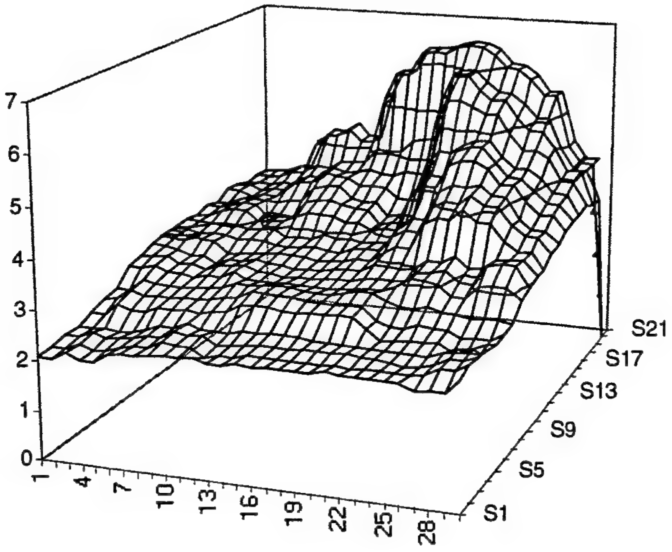


Figure 6.4-5. The value function computed by the Residual Gradient algorithm using a perfect model after 100 trials

6.5 Discussions

We have presented analytical and empirical studies evaluating the relative efficiency of SARSA and residual gradient algorithms using both prediction and stochastic control tasks. These studies provide valuable information for selecting reinforcement learning methods for gaze control for foveal machine vision in real world (commercial, industrial, and military) settings.

These results showed consistent, significant, and sometimes large advantages of SARSA over residual gradient algorithms. They showed that to be useful in a stochastic Markov decision problem, the residual gradient algorithm must use model information, be it learned or known *a priori*. Furthermore, performance degrades with decreasing model accuracy. This bears on the question of the necessity of model information as control problems become non-deterministic. The main theoretical advantage of the residual gradient algorithm is that it converges in many cases to a minimum mean squared Bellman error solution, particularly when function approximators are used. SARSA, on the other hand, does not share this theoretical advantage. In practice, however, this may not be of great significance. All of our experimental results suggest far better performance is obtained with SARSA than with residual gradient algorithms.

Section 7: CONCLUSIONS

Foveal machine vision is inherently active; active vision has two basic components: visual data processing and gaze control. Gaze control for active machine vision must maintain closed-loop system integrity and performance over a wide range of operating conditions. This objective can be difficult to achieve due to several circumstances including the complexity of the performance objectives, the presence of uncertainty, and most of all extremely limited *a priori* model information. Under such conditions, it is very difficult or even impossible to design a control policy with fixed properties that meets the desired performance specifications. This report has addressed the problem of developing reinforcement learning algorithms with maximum computational efficiency as well as effective generalizing capabilities for gaze control in foveal machine vision systems.

The report began by giving a quick introduction to the field of reinforcement learning, a branch of machine learning, and various issues associated with reinforcement learning. It then described a particular theoretical framework within which to explore efficient computation and learning generalization. The description of dynamic programming provides a theoretical basis that serves both to explain the operation of a class of computational procedures, known as TD methods, and to relate them to existing theories of prediction, control, and learning.

The report presented a reinforcement learning method, namely SARSA, in conjunction with recurrent neural networks for gaze control for active perception. The role of recurrent neural networks is to provide a memory mechanism for learning salient history features. The efficacy of the method was empirically validated in a moderately complex active perception problem. And results showed that efficient gaze control for active vision can be achieved using reinforcement learning.

The report then moved on to describe UB's effort in developing constrained Q-learning algorithms for foveal target detection and various techniques, such as learning vector quantization and ATR architecture, to address effective input generalization in reinforcement learning.

Finally, to be commercially feasible, reinforcement learning control methods must be efficient. The report examined two algorithms with function approximation: residual gradient learning and SARSA. The algorithms were evaluated using both prediction and control tasks. SARSA showed consistent and significant superior performance over residual gradient learning. Their relative efficiency in these tasks provides important cues for selecting reinforcement learning algorithms for gaze control in commercial foveal machine vision systems.

Section 8: REFERENCES

Albus, J.S. "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control.* September, 220-227, 1975.

Baird, L. "Residual algorithms: Reinforcement learning with function approximation," *Proceedings of the Twelfth International Conference on Machine Learning,* 1995.

Bandera, C. "Structures and Algorithms for Foveal Machine Vision," Technical Report, Defense Technical Information Center, 1992.

Bandera, C. "Attention Mechanisms and Behaviors for Foveal Machine Vision," Technical Report, Defense Technical Information Center, 1994.

Bandera, C., Vico, F. J., Harmon, M. E., Baird, L. C. III, "Residual Q-Learning Applied to Visual Attention," ICML96.

Barnhill, R.E. "Representation and approximation of surfaces," *Mathematical Software III.* 69-120, 1977.

Barto, A.G., Sutton, R.S., Watkins, C.J.C.H. *Learning and sequential decision making.* (COINS Technical Report 89-95). Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1989.

Bellman, R.E. *Adaptive Control Processes.* Princeton University Press, 1961.

Bertsekas, D.P. "A counter example to temporal-difference learning," *Neural Computation,* Vol. 7, 270-279, 1994.

Bertsekas, D.P. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice Hall, Inc., 1987.

Bertsekas, D.P., Tsitsiklis, J.N. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, Inc., 1989.

Boyan, J., Moore, A. *Generalization in reinforcement learning: Safely approximating the value function. NIPS-7,* 1995.

Boyan, J., Moore, A.W. "Learning evaluation functions for large acyclic domains," *Proceedings of the Thirteen International Conference on Machine Learning*, 1996.

Franke, R. "Scattered data interpolation: Tests of some methods," *Mathematics of Computation 38(157),* 1982.

Fu, K.S., Learning Control Systems– Review and Outlook. *IEEE Transactions on Automatic Control,* 210-221, April 1970.

Girosi, F., Jones, M., Poggio, T., Regularization theory and neural networks architectures. *Neural Computation 7,* 219-269, 1995.

Gordon, G., Stable function approximation in dynamic programming. *Proceedings of the Twelfth International Conference on Machine Learning,* 1995.

Kaelbling, L.P., *Learning in embedded systems.* Ph.D. Dissertation, Dept. of Computer Science, Stanford University, CA 1990.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D., Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation* (1) 541-551, 1989.

Lin, L.J. "Self-improving reactive agents based on reinforcement learning, planning, and teaching," *Machine Learning*. 8(3/4), 293-321, 1992.

Miller, W. T. III, Glanz, F. H., Kraft, L. G. III. "Application of a general learning algorithm to the control of robotic manipulators," *The International Journal of Robotics Research 6(2)*. 84-98, 1987.

Minsky, M.L. "Steps toward artificial intelligence," *Proceedings IRE 49*. 8-30, 1961.

Mitchell, T., Keller, R., Kedar-Cabelli, S. "Explanation-based learning: A unifying view," *Machine Learning*. 1, 47-80, 1986.

Moore, A.W., Atkeson, C.G. "Memory-based reinforcement learning: Converging with less data and less real time," *Proceedings of NIPS '92*, 1992.

Narendra, K.S., Thathatchar, M.A.L. *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice Hall, 1989.

Peng, J. *Efficient dynamic programming based learning for control.* Ph.D. Dissertation, Northeastern University, Boston, MA, 1994.

Peng, J., Williams, R.J. "Incremental multi-step Q-learning," *Machine Learning,*22, 283-290, 1996.

Peng, J., Williams, R.J. "Incremental multi-step Q-learning," *Proceedings of the Eleventh International Conference on Machine Learning,* 226-232, 1994.

Peng, J., Williams, R.J. "Efficient learning and planning within the Dyna framework," *Adaptive Behavior Vol. I No. 4*, 1993.

Powell, M.J.D. "Radial basis functions for multivariable interpolation: A review," In *Algorithms for Approximation.* (Mason, C. \& Cox, M.G., Eds) Clarrendon, Oxford 143-167, 1987.

Ross, S. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.

Schumaker, L.L. "Fitting surfaces to scattered data," *Approximation Theory II.* Academic Press 203-268, 1976.

Rummery, G., Niranjan, M. *On-line Q-learning using connectionist systems*. Technical Report CUED/F-INFENG/TR 166, Cambridge University, 1994.

Schwartz, A. "A reinforcement learning method for maximizing undiscounted rewards," In *Proceedings of the Tenth International Conference on machine Learning*. 298-305, 1993.

Sutton, R.S. *Personal communication*, NSF Reinforcement Learning Workshop, Harpers Ferry, WV, April 12-14, 1996.

Sutton, R.S.. "Generalization in reinforcement learning: successful examples using sparse coarse coding," *Proceedings of NIPS*, 1995.

Sutton, R.S. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," *Proceedings of the Seventh International Conference on Machine Learning*. 216-224, 1990.

Sutton, R.S. "Learning to predict by the methods of temporal differences," *Machine Learning.* 9-44, 1988.

Tstsiklis, J.N., Van Roy, B. *An analysis of temporal-difference learning with function approximation.* LIDS-P-2322, Laboratory for Information and Decision Systems, MIT, March 1996.

Tesauro, G. "Practical issues in temporal difference learning," In J.E.Moody, S.J.Hanson, R.P.Lippmann (Eds.), *Advances in Neural Information Processing Systems 4*}. 259-266, 1992.

Watkins, C.J.C.H. *Learning from delayed rewards*. Ph.D. Dissertation, King's College, UK 1989.

Werbos, P.J. "Consistency of HDP applied to a simple reinforcement learning problem," *Neural Networks*. (3) 179-189, 1990.

Williams, R. J., Peng, J. "Function optimization using connectionist reinforcement learning algorithms," *Connection Science 3(3)*, 1991.

Gaze control for active vision:
Reinforcement learning using modified Q-learning algorithms

by
Peter Scott and Jason Choy
Dept ECE
SUNY at Buffalo
Buffalo NY 14260

Final Report
Item no. 006.3465-901-44 00
Reference: Amherst Systems Inc. PO E3465

Contents:

1. Introduction

Reinforcement learning (RL) is a relatively new strategy for sequential decision making which combines features of the more traditional supervised and unsupervised learning paradigms. Instead of requiring an all-knowing teacher to explicitly define what the ideal action is at each step during training, RL uses an iteratively updated scalar function, called the reward or penalty function, to generate the required feedback for policy iteration. It has been shown that (under reasonably weak conditions) the use of repeated systematic trial-and-error RL training interactions with the environment is sufficient to guarantee that the resulting policy will converge to the overall optimal policy for that task.

Recently RL has received increasing attention in the adaptive control, robotics and artificial intelligence communities. In lengthy multistage optimizations it is often very difficult to specify the optimal action for supervised learning, and to rely on unsupervised training may lead to very slow convergence. In addition, an RL agent can be designed with little or no a priori knowledge about the environment and system dynamic parameters bs the programming effort required is much less than most other learning techniques, yet performance is in many cases comparable to supervised learning. Numerous researchers in robotics have turned to RL as their primary training technique (Asada et al., 1995, 1996a, 1996b)(Fagg. et al., 1994). Their positive results encourage us to investigate the suitability of RL here.

This report reviews the results of the work of the UB group (Peter Scott, Jason Choy) supported by Amherst Systems Inc. under a contract implemented as PO E3465 dated 1-15-96 which is aimed at applying RL technique to the practical problem of target detection from an active visual sensor. . Due to the inherent complexity of this problem it is very difficult, if not impossible, to construct an optimal visual sensor control routine (action policy) directing a robot with an active visual sensor to behave optimally when tasked with target detection. Applying RL schemes to various versions of this problem, our hope is that after adequate training (adaptation of the evaluation and policy maps), the system may be induced to exhibit effective target detection behavior. That as experience accumulates, the decisions made by the system will grow ever more skillful, tending asymptotically to the optimal policy in static, fully observed cases and perhaps in other cases as well. In addition to the prospect of symptotic optimality, the simplicity of an RL agent design lends itself to practical heirarchical multi-agent architectures and multi-tasking environments, important considerations for future work.

This report is structured as follows. Section 2 describes a simplified or "tokenized" version of the general target detection problem from visual data that we have used for numerical experiments. Section 3 describes some non-RL-based approaches to the tokenized problem whose performance measures then serve as baselines for evaluating the results of RL approaches. Section 4 introduces Q-learning, the specific RL technique that was used throughout this report. In Section 5 the experimental results are discussed. Section 6 describes efforts to generalize the input space with artificial neural networks. Finally, Section 7 offers some conclusions and summarizes open problems targeted for future investigation.

2. Problem Description

We focus our study on the target detection problem in active machine vision, with application to robotics. The agent (robot) is equipped with an active vision sensor that it can orient using a given repertoire of control actions. The agent is assumed to be knowledgeable concerning the target, that is, to have access to a full resolution target model, The agent processes a frame of imagery from this sensor, preattentively seeking and marking the most favorable candidate instance of the desired target. A confidence measure is applied to theis candidate "blob." If there is sufficient uncertainty to require further observations before a decision is made (detection of a target declared and its position specified, or the candidate abandoned), the agent sends a request to steer the optical head of the sensor so that the candidate blob is centered in the field of view (FOV). Another frame of data is then acquired and the cycle repeated until an instance of the target is, with acceptable confidence, located. The goal is a sensor feedback control strategy that will detect the target efficiently, i.e. with least cost. Cost may be scored as time-to-target (number of frames of data processed on average before a target is successfully found) , travel-to-target (minimum total angular slew of the optical axis), or some combination of these measures.

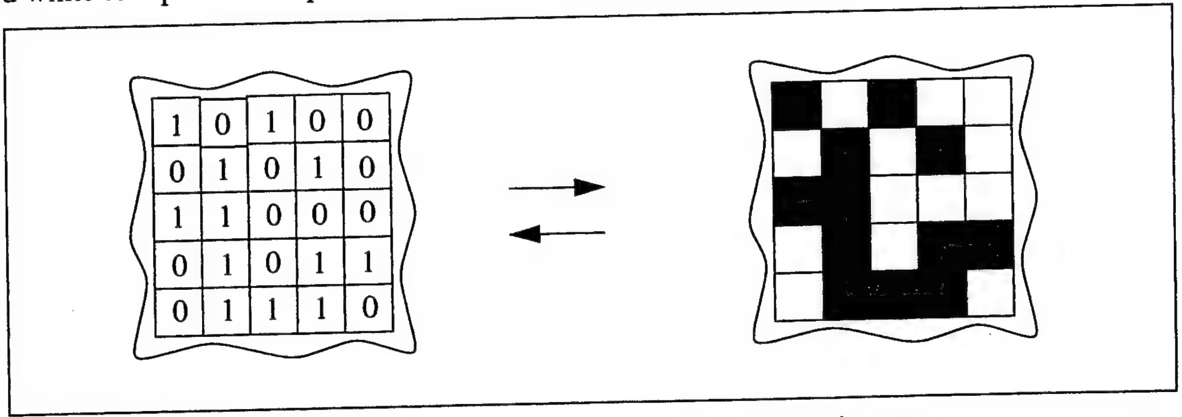
To clarify the underlying issues we utilize a token problem as a test bed for numerical experiments. The token problem is a greatly simplified version of the above general target detection problem that nonetheless retains many of its essential elements. In the simplified environment the basic character of the problem may be better discerned, with secondary issues which may obscure the more central ones removed.

In the token problem, the observed scene is modelled as a a square lattice of binary white noise, the target defined as a simple 3x3 binary pattern, and the control actions limited to a few North-South or East-West movements of the optical axis of the system. The details of this token problem are specified in the following sections.

2.1 Environment

The environment or scene which contains instances of the target that the agent is to search for is a large planar lattice of square pixels. The binary value (0 or 1) of the pixel is the result of repeated Bernoulli trials with equal probabilities P(0) = P(1) = 0.5, all pixel values independent. Thus the whole scene can be described as white noise with binary independent identically distributed (BIID) pixels.

For visualization purpose, we associate the color black to represent the pixel value of 1 and white to represent the pixel value of 0. e.g.



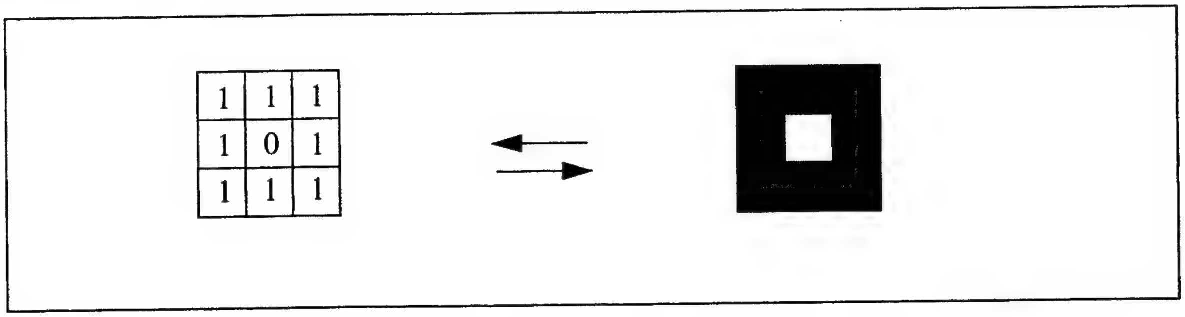
1	0	1	0	0
0	1	0	1	0
1	1	0	0	0
0	1	0	1	1
0	1	1	1	0

Figure 1: Numerical and graphical representation of the environment

Ideally, a totally new scene (environment) should be generated for each trial, however the same effect can also be achievedby placing the sensor at a random location to begin each trial, provided that the environment is large enough.

2.2 Target

The specific target that the agent will search for is defined by the following 3x3 binary template (Fig. 2):



1	1	1
1	0	1
1	1	1

Figure 2: Numerical and graphical representation of the target template

Note that this target has a mean value of 8/9 as opposed to the expected background of 1/2. The difference in these means will be a useful cue when the target is viewed at low resolution. Note also that the targets are not superposed on the background after the background has been generated, they are simply part of the binary IID scene. Thus the probability of a given pixel location being the center of a target is exactly $(1/2)^9$. A 50x50 scene is shown as Fig. 3, with the centers of the four targets found in this scene indicated.

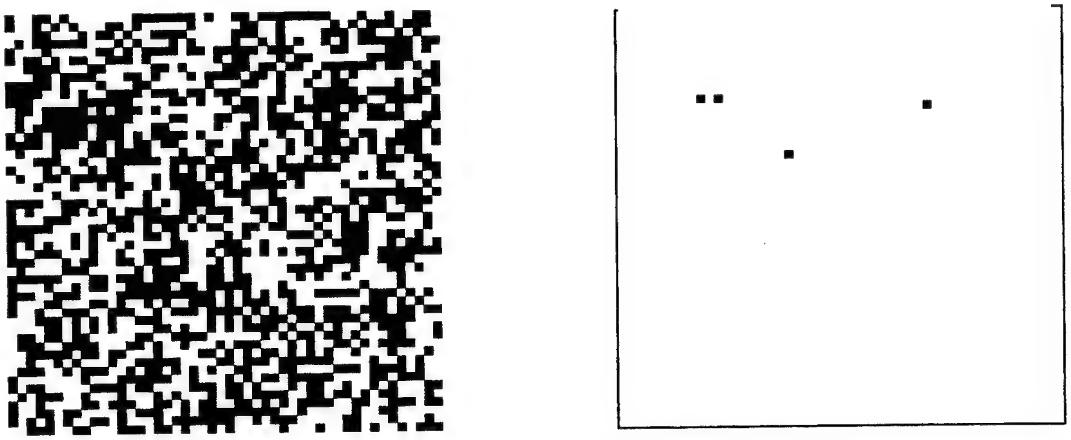


Figure 3: A 50x50 sample environment and its target distribution

2.3 SNR

The signal to noise ratio (SNR) is a significant parameter for target detection. For the problem as described, the signal s_{ij} consists of a random set of targets (as shown in Fig. 2) against a white or 0 background, and the noise n_{ij} is the remainder of the BIID scene x_{ij} as in Fig. 3:

$$x_{ij} = s_{ij} + n_{ij} \tag{1}$$

The mean and second moment of the signal and the noise n_{ij} may be calculated combinatorially. Expanding the expected values for this stationary binary random field,,

$$Es_{ij} = Es_{ij}^2 = 1 \times P(s_{ij}=1) \qquad (2)$$
$$En_{ij} = En_{ij}^2 = 1 \times P(n_{ij}=1)$$

The event $\{s_{ij}=1\}$ that the signal pixel at (i,j) equals one (is black) is the event that the center pixel of a the target happens to be one of the eight nearest neighbors of the point (i,j). Consider a 5x5 array of pixels centered at (i,j). To evaluate the probability of this event we count the number of primitive outcomes associated with it. First consider the number of outcomes consistent with a target northwest (NW) of (i,j). i.e. centered at the location (i-1,j+1). There are $2^{25-9} = 2^{16}$ such distinct binary 5x5 arrays. There are 2^{16} outcomes with a target NE, of which 2^{10} have already been counted (contain both NW and NE targets). Thus the number of primitive outcomes with targest NW or NE is $2^{16} + (2^{16} - 2^{10})$. Continuing, there are 2^{16} outcomes with SE targets, of which 2^{10} have already been counted in each of the two previous cases, so $2^{16} - 2\text{x}2^{10} + 2^4$ primitive outcomes must be added (the last term corrects for counting duplicates twice since there are 2^4 outcomes with targets in all three of the NW, NW and SE corners). The SW case adds $2^{16} - 3\text{x}2^{10} + 3\text{x}2^4 - 1$ primitive outcomes. Adding, there are

$$4\text{x}2^{16} - 4\text{x}2^{10} + 4\text{x}2^4 - 1 = 258{,}111 \qquad (3)$$

primitive outcomes of 5x5 binary arrays with targets to the NW, NE, SE or SW of the central pixel. The set of binary arrays with targets centered N, E, S, or W, can be constructed by reordering the pixel locations of the set of those centered NW, NE, SE and SW, so the total number of primitive outcomes with a target centered at one of the eight nearest neighbors of (i,j) is double the above, and

$$P(s_{ij}=1) = (2 \times 258{,}111) / 2^{25} = .01538 \qquad (4)$$

Note that the sample environment depicted in Fig. 3 is consistent with the value, since the four target instances found comprise 29 black pixels out of 2500. Substituting (4) into (2) we compute the SNR to be (5a,b)

$$SNR = \frac{\sigma_s^2}{\sigma_s^2} = \frac{Es_{ij}^2 - (Es_{ij})^2}{En_{ij}^2 - (En_{ij})^2}$$

$$= \frac{01538 - 01538^2}{0.5 - 0.5^2} = 0.06057$$

Thus the SNR in dBs is 10 log 0.06057 = -12.2 dB for this target detection problem. For many target detection methods this is a challengingly low SNR, with target equivalent power only 6% of the noise background equivalent power.

2.4 Sensors

We experimented with both uniform resolution and multi-resolution sensors. The former contains a 3x3 array of sensor lattice sites (pixels) as shown in Fig. 4, the latter consisted of a 3x3 fovea surrounded by a ring of resolution cells or "rexels" each 9 pixels large as depicted in Fig. 5.

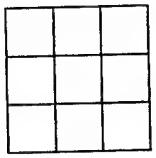


Figure 4: 3 x 3 Uniform-Resolution Sensor

For the foveal sensor case, a simplified version of the unsubdivided exponential foveal sensor with radix 3 (Bandera 90) is used. The sensor has two regions with different resolutions. The inner zone or fovea is exactly the same as the uniform-resolution sensor described above, while the outer ring contains four rexels with receptive fields each the same size as the entire 3x3 pixel fovea. Each of the rexels produces a single output value, which can equivalently be the sum or the average of the 9 pixel values within its receptive field.

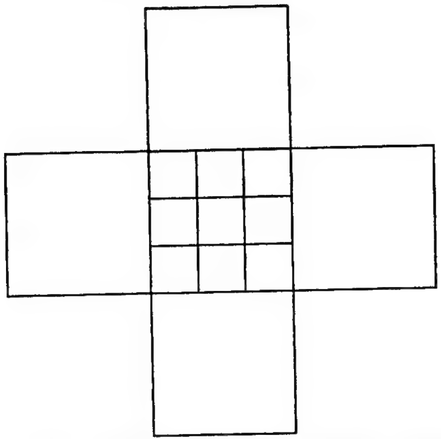


Figure 5: 2-ringed undivided exponential foveal sensor (radix 3) with partial coverage of outer ring

Note that the presence of a target exactly registered in one of the low-resolution rexels outside the fovea is signalled by a high rexel value, 8 out of a maximum of 9 (or 8/9 if the average rather than total value is reported by these rexels). This value does not guarantee target present, however, since there are 8 non-target patterns that would yield the same rexel value. While a rexel of value 8 has only probability 1/9 of containing a registered target, that is considerably more than the a priori probability of 1/512 and makes such a rexel a good candidate for subsequent higher-resolution investigation. In general the darker the rexel, the more likely it contains all or part of a target (up to the "pure-black" rexel value 9, which cannot contain a target or even a large fraction of a target).

2.5 Control Action Set

Three different sets of control action were used to simulate the agent's capacity to steer the visual sensor. These actions translate the fixation point, the location in the scene imaged at the center of the fovea, as follows:

Set A: Move N, E, S or W a distance of 1-3 pixels.
Set B: Move N, E, S or W a distance of 1-5 pixels.
Set C: Move along the 8 principal compass directions 1-3 pixels

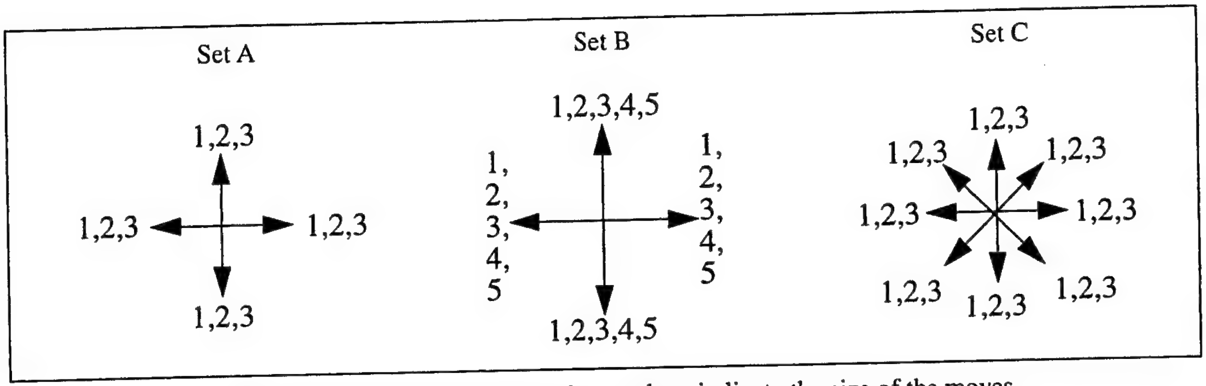


Figure 6: Control Action Sets, the numbers indicate the size of the moves

2.6 Cost function

There is a cost associated with each action that the agent performs. The goal is to minimize the discounted aggregate cost to first detection of an instance of the target. In this study, the cost is defined by the following aggregate cost function (performance index) (6):

$$J = \sum_{i=0}^{n-1} \gamma^i (C_1 |a(i)| + C_2)$$

where a(i) is the action taken at the ith step, |a(i)| is the distance moved by the fixation point in pixels, C_1 and C_2 are constantsdefining the relative importance of distance and time, and n is the number of timesteps to first target detection. This performance index reflects our interest in minimizing time to target acquisition. The first term in parenteses in (6) penalizes proportionally to the distance moved by the fixation point, or slew of the optical axis of the sensor, since larger slew takes more time. The second term is associated with the constant dwell time once the fixation point has been reached, and the subsequent constant time to compute the next action selection before the next action can be initiated. The stepwise cost will be used as penalty in the RL algorithm while a fixed scalar value will be used as the reward for detecting the target.

2.7 Operation

In these numerical experiments, the agent is assumed to have knowledge of the target template (Fig.2), can move its visual sensor according to the action set selected (Fig.6) . It can also register the costs (6) associated with the actions it performs.

Initially, the agent's visual sensor is aligned to a random location in the scene. The agent receives a frame of data from the sensor, and selects an action. Upon execution of that action and receipt of the next frame of data, either a penalty or a scalar reward value will be received depending on whether a target has been registered. This perception-action sequence continues until a target is detected or the search is declared terminated with failure due to hitting hard time or space bounds. In any of these cases, the trial is declared complete and its results recorded.

The control policy (action selection strategy) will be modified and improved as the agent continues interacting with the environment. The goal of the agent is to converge toward an optimal policy minimizing the cost function .

3. Reference policies

Before considering reinforcement learning approaches to this problem, a set of comparison or reference policies not based on learning are described. The performance of these policies will serve as a basis of comparison, a yardstick by which to measure effectiveness of the reinformcement learning approaches..

3.1 Randomized policies

Three different policies incorporating random selection of actions from among a prescribed set of permissible actions were simulated. The first allows unconstrained choice of action within the selected action set A, B or C described in Section 2.5. The other two incorporate memory of past perceptions to constrain the choice, preventing actions from being selected which are guaranteed not to find a target.

3.1.1 Unconstrained random policy

To establish a performance floor, random walk experiments using the action sets A, B and C described in Secition 2.5 , with a uniform resolution 3x3 sensor, were performed. For each trial, the sensor is initially fixated at a random location in the environment and the agent randomly selects an action from the selected action set with uniform probabilities, repeating until the target is detected (registered with the sensor) or the limit of saccades is reached. When a target is found or the saccade limit reached, the trial is complete. The results of 100 trails averaged for each of the action sets is slown below.

Table 1: Unconstrained random policy performance

Results of 100 Trials	Distance to target (pixels)	Time to target (fixations)
Set A	1236.77	616.90

Table 1: Unconstrained random policy performance

Results of 100 Trials	Distance to target (pixels)	Time to target (fixations)
Set B	1351.82	450.28
Set C	1182.20	589.62

Since the probability of blindly landing on a target is 1/512, the 450-600 fixation performance of the unconstrained random policy is roughly as expected for blind action choices. Averge performance worse than 512 fixations is largely due to revisits, that is moving back to pixel locations already determined not to center a target. Fixation point trajectories for 500 step random walks are shown in Figure 7. Note that since Set A is smallest action set,, the likelihood of revisiting fixation points already visited is higher than the other sets, and its time to taret performance is poorer than for the larger action sets. Such revisits are of course wasted observations, and can be reduced by the inclusion of memory into the control action selection process.

r

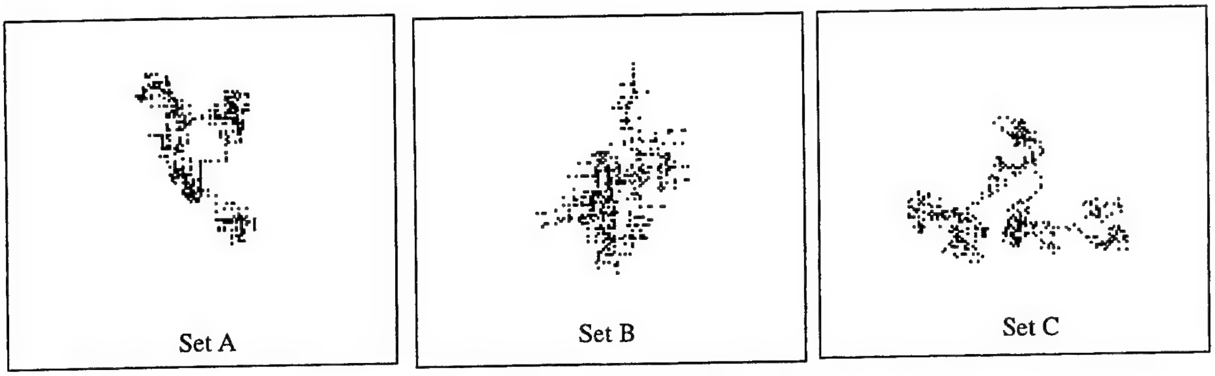


Figure 7: Trace (path) of a sample 500-move random-walk experiment generated by the action sets. Set A resulted in 374 distinct visited locations, Set B resulted in 418 distinct visited locations, Set C resulted in 416 distinct visited locations. Set A issomewhatmore likely to re-visit locations already visited.

3.1.2 Constrained random approaches

The ideal way to prevent revisits and related inefficient actions is to recall all perceptual data to date. In this perfect-memory case, regions with have already been detemined to have no chance of containing a target will never again be fixated. Perfect recall is assumed in the greedy-immediate policy to be described in Section 3.2. In many applications, however, perfect recall may demand unacceptably large memory. Two policies based on remembering limited features of the fixation trajectory are described. Each limits revisits by retaining some useful information about where the sensor has been.

Direction-register contraint: Before selecting a control action, the agent notes the direction of the previous action, and all those actions leading back are banned.This is a minimal memory burden, requiring a single two-bit (Cases A, B) or three-bit (Case C) register.

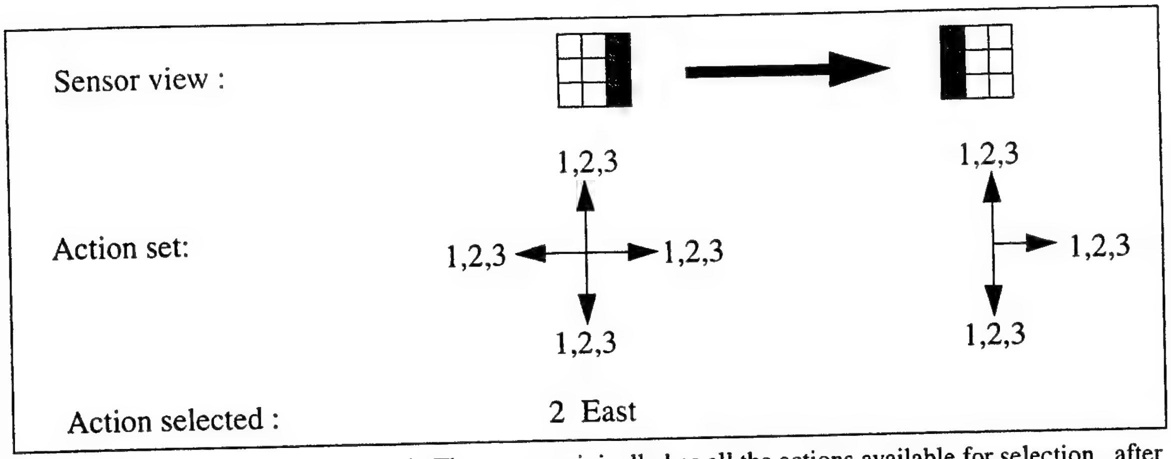
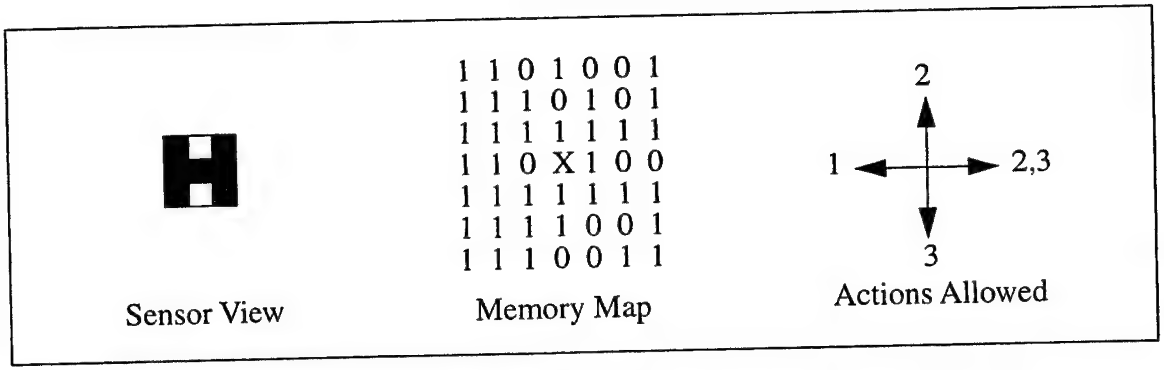


Figure 8: Direction-Register Approach. The sensor originally has all the actions available for selection , after it moved 2 units East, all the actions leading back (West actions) are banned on the next step.

Fixation-window constraint: The agent keeps a WxW bit sliding window centered at the current fixation point and marking locations of previous fixation points within the area of the window. Actions landing on previously visited pixels are banned. Locations of any fixation points outside the current window are lost as the window slides. The memory burden for this approach is W^2-1 bits..



```
1 1 0 1 0 0 1
1 1 1 0 1 0 1
1 1 1 1 1 1 1
1 1 0 X 1 0 0
1 1 1 1 1 1 1
1 1 1 1 0 0 1
1 1 1 0 0 1 1
```

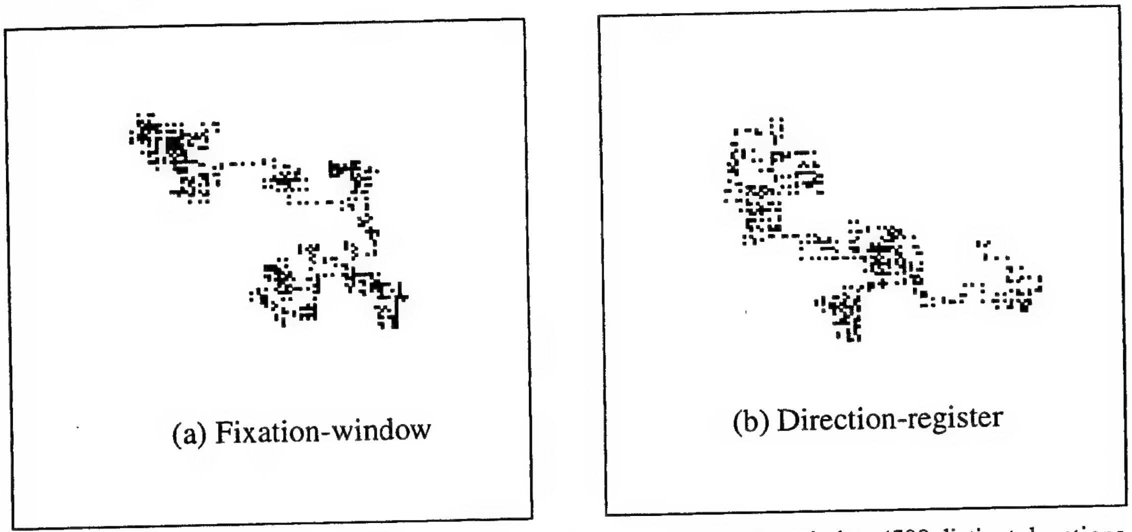
Figure 9: fixation-window Approach. "1" in the fixation-window indicates a previous fixation point. tion has been visited before. All those actions that lead to a visited location will be banned from selection.

Note that the fixation-window data is not sufficient to reconstruct the perceptual data previously observed from that region. To retain this would require three rather than two values per pixel (0, 1, unknown). While perceptual memory would be somewhat more informative than fixation point memory, the addition of 50% more memory was not considered to be justified.

The simplest way to use these constraints is to select the next action at random from the constrained set of choices. Results are summarized in Table 2. As anticipated, the more memory-intensive fixation-window approach is superior in both distance and time metrics. Typical fixation trajectories for 500 steps of the the direction-register and fixation-window approaches are shown in Fig. 10. Comparing to Fig. 7, both explore more territory than the unconstrained random approach. The Fixation-window trajectory visits 500 locations, the direction-register visits 450, ie. has 50 revisits to none for the fixation-window apporach.

Table 2: Performance of the random approaches

100 trials (Set A)	Distance to target (pixels)	Time to target (fixations)
Unconstrained	1236.77	616.90
Direction-register	1009.94	504.17
Fixation-window (7x7)	982.20	487.98


(a) Fixation-window

(b) Direction-register

Figure 10: Fixation trajectories for 500 fixations: (a) Action set A with fixation window (500 distinct locations) (b) Action set A with direction-register approach (450 distinct locations visited)

3.2 The greedy-immediate policy

A final, and better, non-learning-based policy was employed in order to provide a stringent baseline against which to evaluate the performance of the reinforcement learning strategies. The policy is based on perfect recall of all past perceptual data. Actions are chosen to maximize the true probability of immediate one-step target detection, with no exploration, and this policy is therefore called the *greedy-immediate policy.* It is not overall-optimal, since in certain positions it leads to larger expected time or distance to target detection than the overall best action. While not overall-optimal, its performance is judged to be near-optimal, unlikely to be greatly inferior to the exact but unknown true optimal policy.

The greedy-immediate policy assumes the agent possesses unlimited memory, remembering all past perceptions exactly. Given a perception, for every pixel (i,j) in the entire scene, the value of $P_{TC}(i,j)$, the probability that pixel is a target center given the complete past sequence of perceptions, is evaluated. Initially all such probabilities are set at 2^{-9} since all 512 3x3 binary patters are equally likely everywhere. Upon each new perception, after all pixel target probabilities are updated, the agent will choose the action that will saccade to the fixation point with the maximum probability of being a target center amon all pixels reachable in a single move. If there are several such best one-step candidate fixation points, one is selected at random.

Define the set of pixels Y={(i,j) } contained in a given perception, augmented by their immediate nearest neighbors, as the update set U = { (i+k,j+l): (i,j) ε Y; |k|, |l| < 2}. Only pixels (i,j) ε U require updating, since the new perception contains no data relevant to the remaining $P_{TC}(i,j)$ values. For each (i,j) ε U the probability $P_{TC}(i,j)$ that pixel location (i,j) is the center of a target is updated as follows.

Let $P_B(i,j)$ be the probability that pixel (i,j) is black and $P_W(i,j) = 1-P_B(i,j)$ the probability it is white. After a given perception, every pixel in the scene is in one of three states: known to be black $P_B(i,j)=1$, known to be white $P_W(i,j) = 1$ or as yet unobserved $P_B(i,j)=P_W(i,j)=1/2$.. For each (i,j) ε U compute $P_{TC}(i,j)$ as the nine-way product of $P_W(i,j)$ and $\{P_B(i+k,j+l); |k|, |l| < 2\}$. Thus for instance if (i,j) is known to be black, or any of the eight-nearest-neighbors of (i,j) are known to be white, the location (i,j) cannot be a target center and $P_{TC}(i,j)=0$. In general if m of the nine pixels centered at (i,j) are known to be of the right color for a target (white for (i,j), black for the remaining pixels), none of the wrong color, and the remaining ones unknown, $P_{TC}(i,j) = 2^{m-9}$.

The performance of the greedy-immediate policy was explored in a second set of numerical experiments. Action set A (move 1, 2 or 3 pixels N, E, S or W) was selected for these experiments. Two different perception geometries were simulated: a small 3X3-pixel uniform resolution sensor, and a wide FOV sensor containing five such arrays in the cross pattern shown in Figure 8.

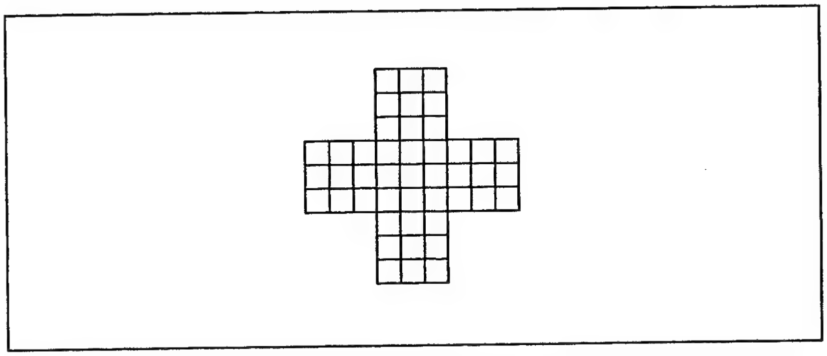


Figure 11 : Wide FOV uniform resolution sensor geometry

Table 3: Greedy-immediate policy performance

Results of 100 Trials	Distance to target (pixels)	Time to target (fixations)
3X3 uniform resolution sensor	336.16	168.94
Wide FOV sensor	338.48	167.79

Comparing Table 2 and Table 3, as expected the greedy-immediate policy performs much better than the others, and the unconstrained random policy much worse. Indeed, these two non-learning-based heuristic policies, one a random policy totally lacking in goal-directed strategy, and the other the presumably near-optimal, may be anticipated to bracket the performance of any interesting learning algorithm. Two reinformcement learning algorithms will be introduced in Section 4 of this report. Their performance will be compared to these reference policies in Section 5.

4. Q-Learning

Q-Learning, a widely used RL algorithm, is briefly reviewed in this section. Interested readers can refer to Watkins 1989 and Watkins and Dayan 1992 for more details. Kaelbling et al 1996 give a comprehensive review of the subject. Some issues regarding the application of Q-leaning to the token problem are also discussed in this section.

It is assumed that the agent (robot) can discriminate a set *S* of distinct world states, and can execute a set *A* of actions on the world. The world is modeled as a Markov process, which makes the next-state probability density function depend only upon the current state and current action taken. . Let *P(s, a, s')* be the one-step state transition probability, the probability that the world will transit to the next state *s'* from the current state *s* when the action *a* is executed.. For each state-action pair *(s, a)*, a reward *r(s, a)* is defined which can be positive for reinforcing the action taken or negative for penalizing.

The general RL problem is typically stated as constructing an optimal policy, one that maximizes the discounted sum of the current plus all future rewards. A policy π is a mapping, deterministic or stochastic from *S* to *A* which maps states into actions to be chosen from those states. The cumulative reward is referred as the *return* and is defined as (7)

$$\sum_{n=0}^{\infty} \gamma^n r_t$$

where r_t is the reward received at step *t* given that the agent started at state *s* and executed action according to policy π. γ is the discounting factor, it controls to what degree future rewards influence the value of a policy, where 0 .

Given the definitions of the transition probabilities and the reward distribution, the optimal policy can be solved using methods from dynamic programming.(Bellman, 1957). It is Q-leaning that makes possible to simultaneously learn the dynamics of the environment and construct the policy.

Let *Q(s, a)* be the expected return or action-value function for taking action *a* in state *s* and continuing thereafter with the optimal policy. It can be recursively defined as: (8)

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in S} P(s, a, s') \max_{a' \in A} Q(s', a')$$

Estimates of the Q values are constructed incrementally on line. *Q(s, a)* is usually start with 0, and when an action is taken, update the Q-values as follows: (9)

$$Q(s, a) \Leftarrow (1 - \alpha)Q(s, a) + \alpha(r(s, a) + \gamma \max_{a' \in A} Q(s', a'))$$

where r is the actual reward value received for taking action a in a situation s, s' is the next state, and α is the learning rate (between 0 and 1).

4.1 Action set and perception-state space

One of the sufficient conditions to guarantee convergence of Q-learning to the optimal policy is to have all state-action pairs used unboundedly often as training proceeds. Thus the size of both the action set and state space play important roles. The product of their cardinalities determine the efficiency, perhaps even the feasibility, of Q-learning approaches.

First consider the action sets. In an effort to minimize the number of available actions without severly constraining performance, action set A from section 2.5 is selected for further investigation. It is smaller than the other action sets tried yet performs almost as well.

Turning to the state space, if the target search is modelled as a Markov process, the underlying state space must be defined as the complete set of probability density functions for all pixels in the entire environment. This constitutes an unworkably large memory burden for realistic problems. The most convenient candidate for the state process, the sequence of frames of visual data acquired by the vision sensor, does not constitute a Markov process. That is, the information that the robot obtains about the environment with each frame is not a complete state of the world for purpose of target detection. If it were, then with s the current frame and s' one possible next frame, P(s,a,s') would be independent of past frames. This is clearly not the case since past frames may overlap with s' and thus P(s,a,s' | S), where S is the entire past sequence of frames, will in general differ from P(s,a,s').

. However, if practical necessity persuades us to abandon guarantees of convergence based upon the Markovian assumption, a partial state, or "perception-state" space can be constructed directly from the information from the sensor. For the 3x3 uniform resolution sensor, two perception-state spaces are proposed. The first one, designated *frame space,* is constructed directly from the current frame of visual data, and there will be 512 (2^9) states in this state space.

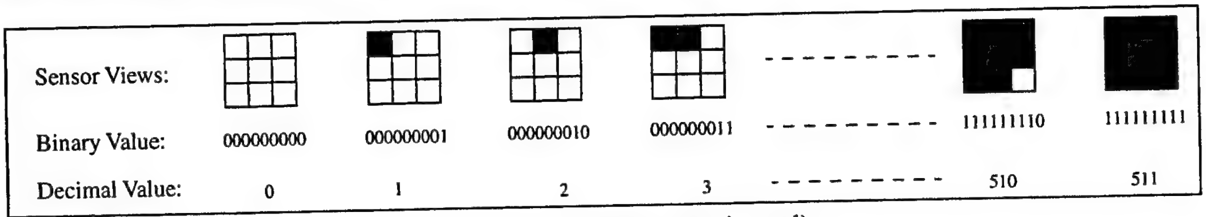


Figure 11: Representation of the frame space (512 possible states in total)

The second perception-state space for the 3x3 sensor is constructed by extracting and then enumerating certain features of the current frame, and the number of these perception-states is much less than that for frame space. This space will be referred to as *feature space.* It is assumed that the agent can preprocess perceptual data to extract features. Specifically, a search is conducted along the 4 compass directions to determine if there is a 1/3 partial match, 2/3 partial match or no match of the target template. These four values consititute the feature set. Since there are 3 possibilities for each of the 4 directions, the total number of states in feature space is 3^4 = 81.

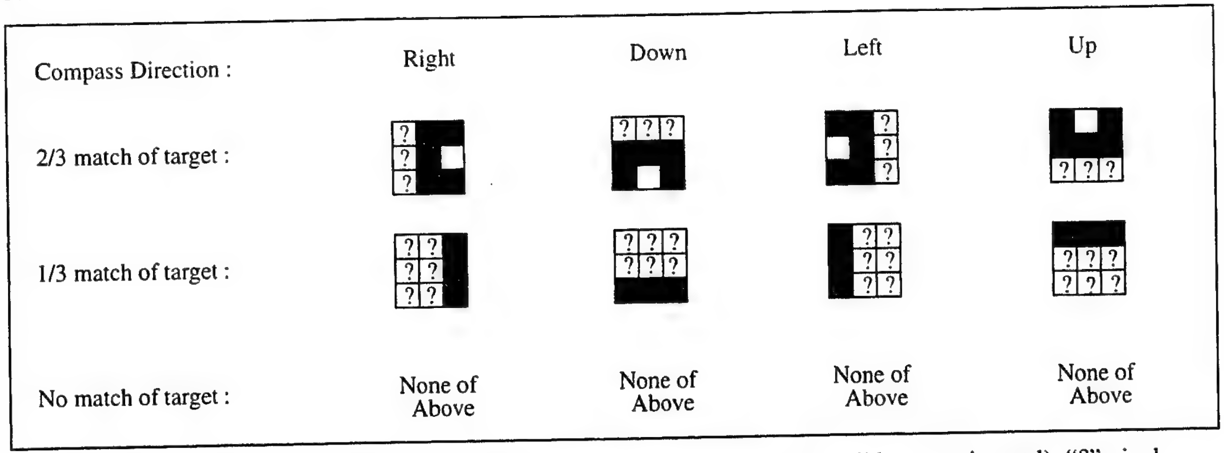


Figure 12: Feature components for the construction of the feature space (81 possible states in total). "?" pixel stands for don't care condition.

A modified version of the wide FOV sensor shown previously in Fig. 8 was also considered. In order to determine the value of having a lower-resolution set of rexels around the fovea, the nine pixels in each of the four peripheral 3x3 regions are combined to a single rexel outputting a single value corresponding to the number of black pixels within its scope. The resulting sensor geometry is shown in Fig. 13 below.

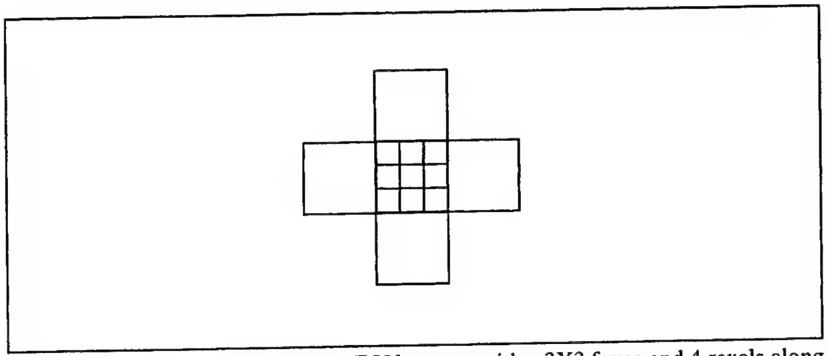


Figure 13.: Modified version of the wide FOV sensor with a 3X3 fovea and 4 rexels along the four principal compass directions.

Note that the original wide FOV sensor has 2^{45} =3.52x10^{13} distinct states, too many rows for a practical Q-value look-up table.The modified sensor of Fig. 13 has 10 distinct values 0-9 for each of the four peripheral rexels and 2 values for each of the 9 foveal pixels for a total of 2^9x10^4 = 5.12x10^5 states (rows), a reduction of more than seven orders of magnitude, but still too many. A perception-state space for this sensor which is similar to the feature space described previously is constructed as follows:

1) The 9 foveal pixels are preprocessed to determine the degree of partial match to the target in the N, E, S and W compass directions. The direction and magnitude (0, 1/3 match, 2/3 match) of the two best matches are stored in order as the foveal substate. 29 distinguishable

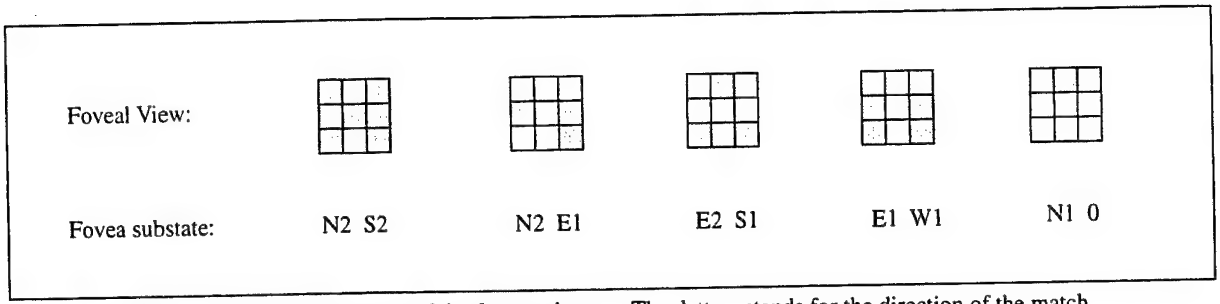


Figure 14. : Examples of some of the fovea substates. The letters stands for the direction of the match the number stands for the match degree, (2)/3 match, (1)/3 match and 0 for no match.

foveal substates are obtained in this fashion. Examples of these substates are shown in Fig. 14.

2) The peripheral rexel values are quantized into the following four sets depending upon their value (number of black pixels):

7-9	--->	D (very Dark)
5-6	--->	d (dark)
3-4	--->	l (light)
0-2	--->	L (very Light

The quantized values of the two rexels located in the directions of the two best foveal matches included in step 1 above are appended to the foveal substates to complete the construction of the perception-state. An example is seen in Fig. 15 below.

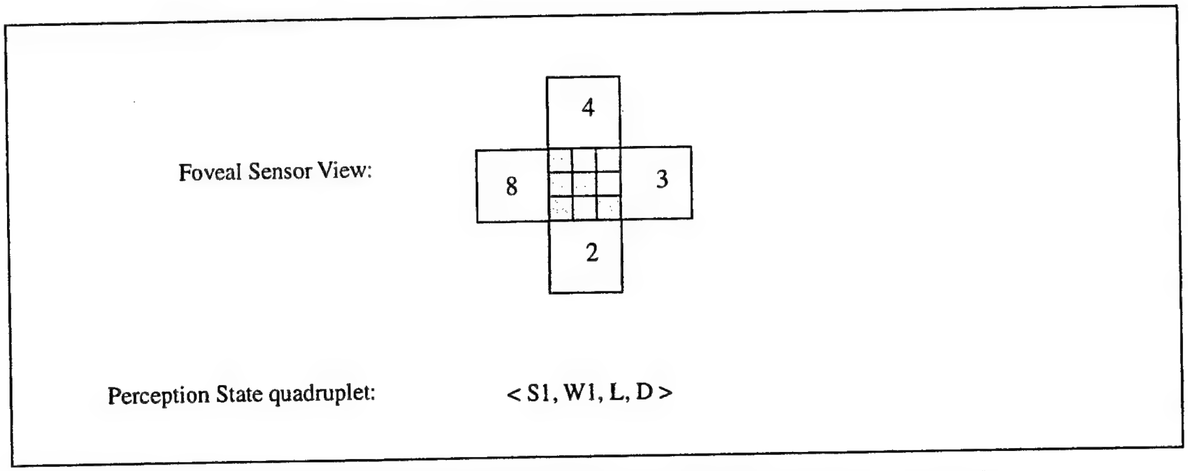


Figure 15 : Example of a perception-state. The number in each rexel represents its value.

Since there are 4 quanta for each of the two corresponding rexels, thus the total number of distinct perception-states is 29 x 4 x 4 = 464. This is a manageable number of rows for a Q-value look-up table.

This mapping is certainly lossy, with the 5.12×10^5 distinct states supplying more infomration of use to target-seeking than the 464 perception-states. Previous experiments showed, however, that effective actions tend to move the fixation point in the direction of best match, that is, the most likely direction to find a near-by target. Knowledge of the rexel value in such a direction is sometimes useful, for instance if there is a 2/3 foveal match in the N direction but the rexel in that direction has a value of 0, 1 or 2 there cannot be a target found by moving one pixel to the north. Most of the time the rexel value will be light or dark, but not very light or very dark, and thus yield a weak cue only. Thus it is anticipated that the performance of this sensor will be some improvement over the performance of the 3x3 sensor not perceiving this cue, but not a great improvement.

4.2 Reward and Discount Factor γ

Feedback signals are generated as soon as the agent takes an action. A fixed positive reward value is assigned whenever the agent has the target template completely in the FOV of the sensor. On the other hand, a negative reward associated with the cost function described in section 2.6 will be applied if the current action has not resulted in a target detection.

The discount factor γ is used to determine to what degree rewards in the future should effect the current policy choices. Usually this value will be in the range [0,1] (e.g. 0.8) for most RL learning experiments. In this token problem, due to the unavailability of the true state, the task of target detection based purely on sensor readings is more similar to a reflexive behavior than a goal-directed behavior along a path: the reflexive behaviors make quick reactive actions with little planning ahead in response to the stimulus(information) perceived. A discount factor γ with much smaller value (0.1 - 0.2) is used for the learning of this kind of reflexive behavior so that the current action-value will be affected less by the distant future rewards and more by the near future rewards.

4.3 Action Selection

The Q-learning algorithm selects the action *a* from perception-state *s* with the best Q-value *Q(s,a)*. In order for sufficient exploration of all the state-action pairs and to avoid trapping inside infinitely repeated inferior action loops without trying new policy choices, the actions in the reported experiments are selected probablilistically based on Q-values using a Boltzmann distribution. Given a perception-state *s*, the corresponding action *a* is chosen with the probability:

$$P(a) = \frac{e^{Q(s,a)/T}}{\sum_{a \in A} e^{Q(s,a)/T}} \qquad (10)$$

where *A* is the set of all available actions (Action set A in this token problem) and *T* is the computational temperature parameter that controls the tradeoff between exploitation and exploration. The action with the largest Q-value is more likely to be selected, while actions other than that one also have a chance to be selected. *T* is decreased over time by a cool-down function as exploration gives way to exploitation.

4.4 Makovian Property

The primary condition for Q-learning algorithm to converge to the optimal policy is a Markovian environment. In other words, the next state of the environment should be determined only by the current state and the action taken. In this environment, all the information needed to determine the current optimal action is reflected in the current state representation.

In the token problem, the above Markov assumption is violated: the sensor fails to make essential distinctions among world states, and the perception-state representation is based purely on the sensor's immediate perception. This violation is referred as the *perceptual aliasing* (Whitehead & Ballard, 1991) and the resulting model is called a *partially observable Markov decision process* or POMDP.

Q-learning cannot be guaranteed to converge to the optimal policy in a non-Markovian environment. However, this does not imply the method is necessarily inappropriate in such environments. Q-learning might still produce a policy with good, if not optimal, performance within a reasonable number of trials for the task of target detection. It is shown later in this report that the use of a small value γ, stochastic action-selection and the application of a small amount of past-perception memory can help to achieve reasonable learning performance.

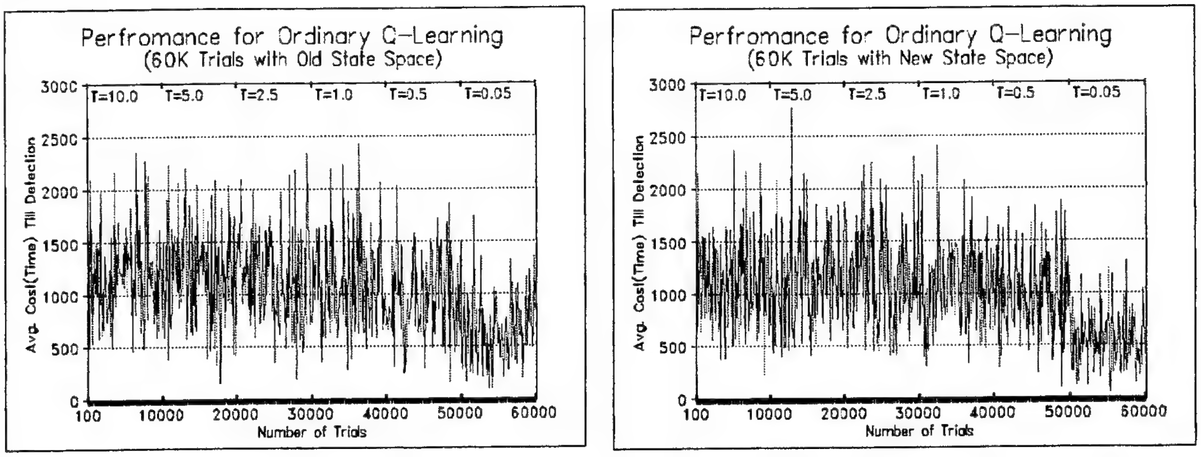
5. Experiments

The experiments divide into two groups. The first uses standard Q-learning. The second brings in the fixation-window approach described in Section 3.1.2 to assist in the process of action selection. Results from both approaches will be compared. Moreover, both perception-state spaces (frame space and feature space) will be used to better understand learning efficiency in the current environment.

The performance is measured by averaging the total penalty for each trial. The miss rate is calculated by counting the number of trials which failed to locate a valid instance of the target within 2000 frames for this experiment together with all those trials whose fixation trajectories crossed the spatial limits of the environment. Both time to target and miss rate indexes are tabulated.

5.1 Standard Q-Learning

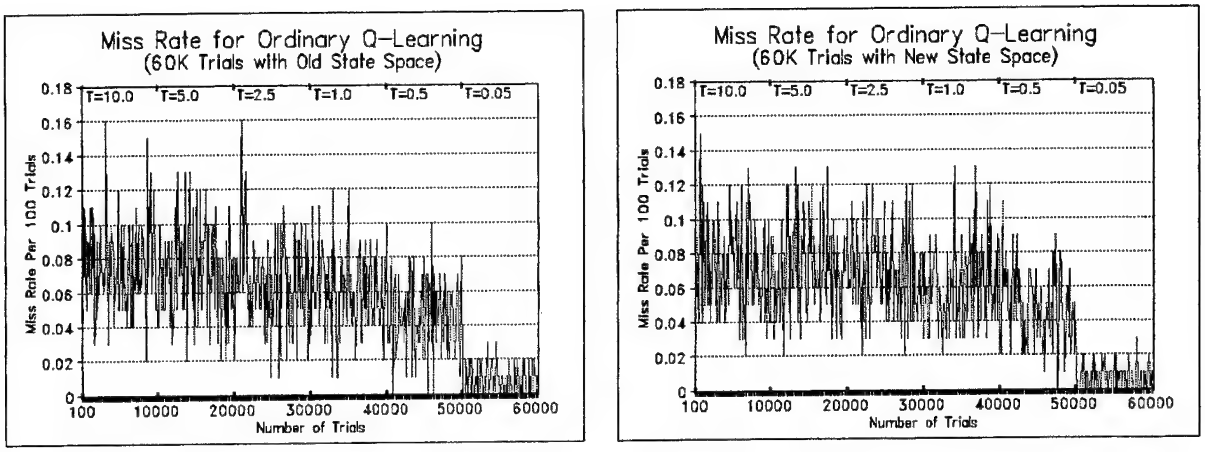
Standard Q-learning was first employed, using both frame and feature spaces.A relatively large number of trials (60,000) were used for training to guarantee adequate exploration of the state-action pairs. In this setup there are 512 x 12 Q-value table entries for the frame space runs, and 81 x 12 in the feature space case. The computational temperature T tabulated on the top of each chart) is decreased step-wise as indicated. The results from the experiments are summarized in the following figures:



(a)

(b)

Figure 16: Performance (average cost J over10 trials by standardard Q-learning for 60K trials - (a)frame space (b) feature space.



(a)

(b)

Figure 17: Miss rate by ordinary Q-Learning for 60K trials with (a) frame space (b) feature space.

It is seen that results from experiments using the two sperception-state spaces produce similar performance after learning.

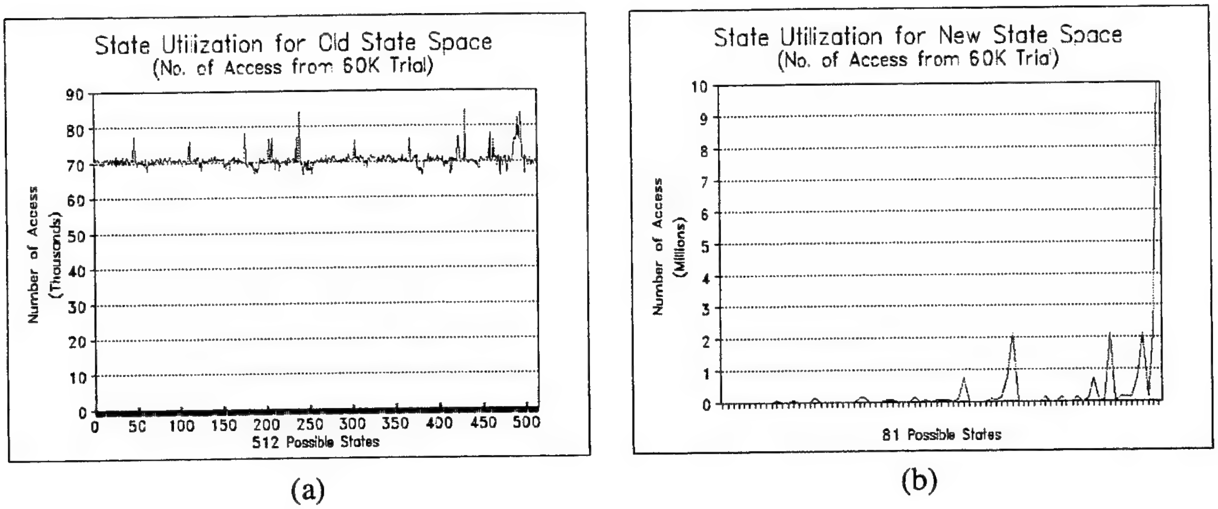


Figure 18: State utilization (number of access stof the states) for (a) Frame space .All of the 512 states are utilized) (b) Feature space. Only 34 out of the 81 states are utilized, state 81 registered the highest number of access since it represent the no-match condition for all directions.

The resulted policies (when T=0.05) demonstrate an improved performance over unconstrained random behavior (when T=10), the mean total time is reduced from around 1000 steps to around 500 steps. On the other hand, due to the dynamic environments (a new scene for each trial) and the violation of the Markov assumption, Q-learning seems slow to converge: the resulting policy can have a performance of 100 steps for some cases while as much as around 1500 steps for others. The same event is observed from the plotting of the miss rate: though the agent is more capable of detecting targets (after learning mean miss rate reduced from around 0.07 to around 0.01), the resulting policy also saw the miss rate vary between 0 and 0.02

It is seen in Fig. 18 that the states in frame space are accessed at about the same level, this is mainly because the binary scene pixel values are generated with equal probability (BIID) . On the contrary, only 34 out of the 81 possible states from the feature space are accessed at all, which reflects the mutual exclusivities among the features for building up the state space and some feature states do no exist(e.g. Fig 19). It is also interesting to note that though 34 state space is a significant reduction from 512 states characterizing the frame space, the performance is about the same. This implies the possibility of generalizing the input state space and reducing the required learning period. To test this possibility, the experiments were performed again with a much smaller number of trials(2,400 trials).

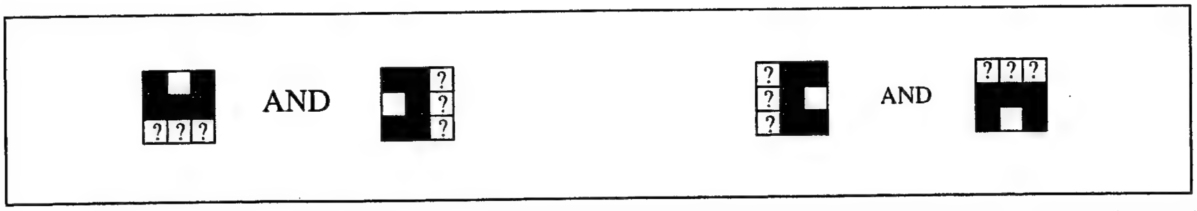
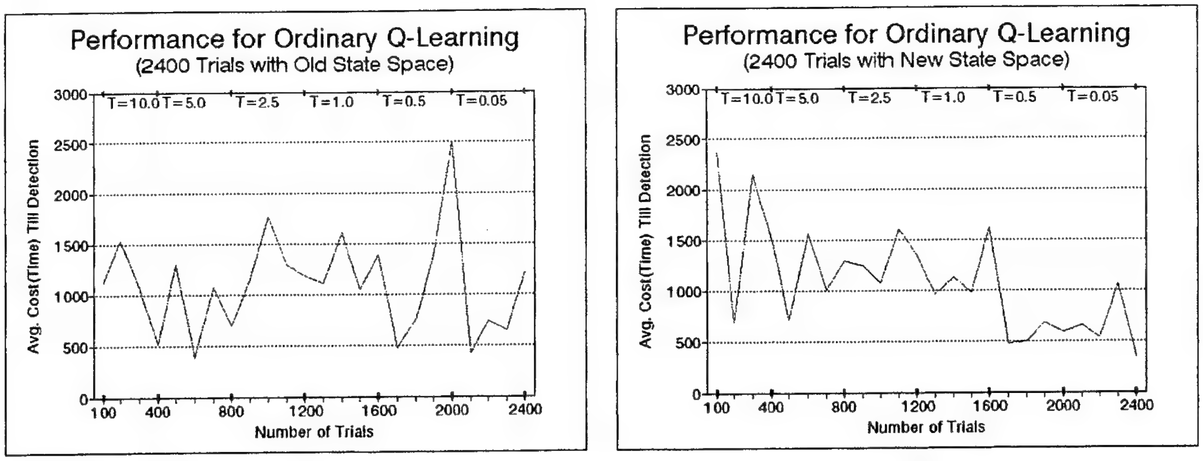
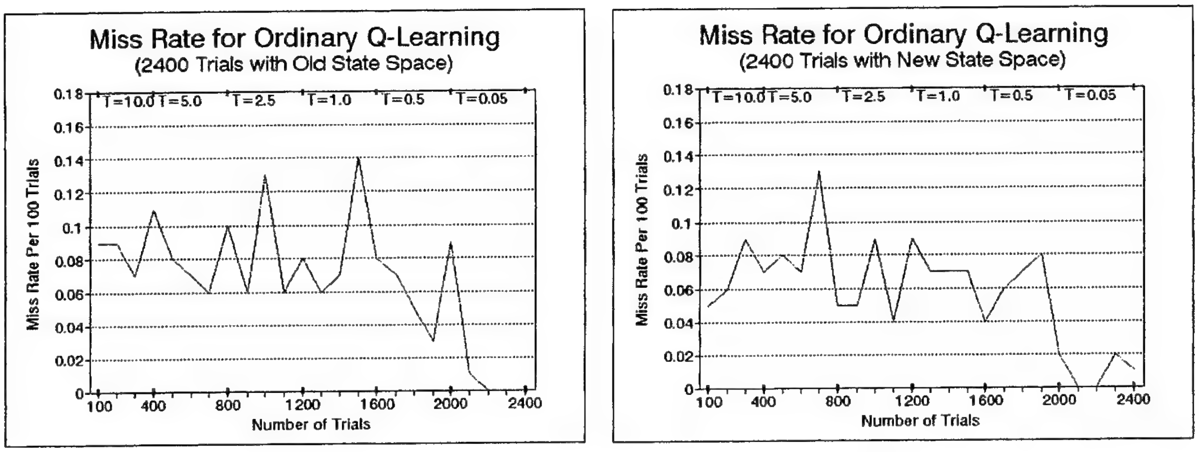


Figure 19: Examples for conflicted features: states composed from these conflicted pairings are impossible to exist thus are not accessed during the trials. The. "?" stands for don't care condition.



(a) (b)

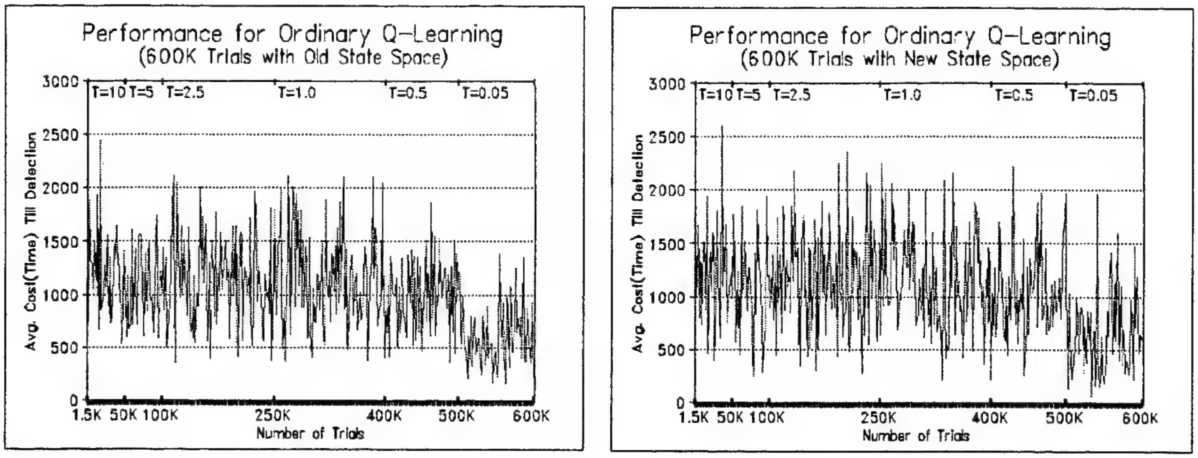
Figure 20: Performance using standard Q-learning with (a)frame space (b) feature space. Lack of a decrasing trend in frame space with T indicates inadequate exploration.



(a) (b)

Figure 21: Miss rate by standard Q-Learning for 2400 trials with (a) frame space (b) feature space. Note good performance by both even in face of inadequate exploration in frame space case.

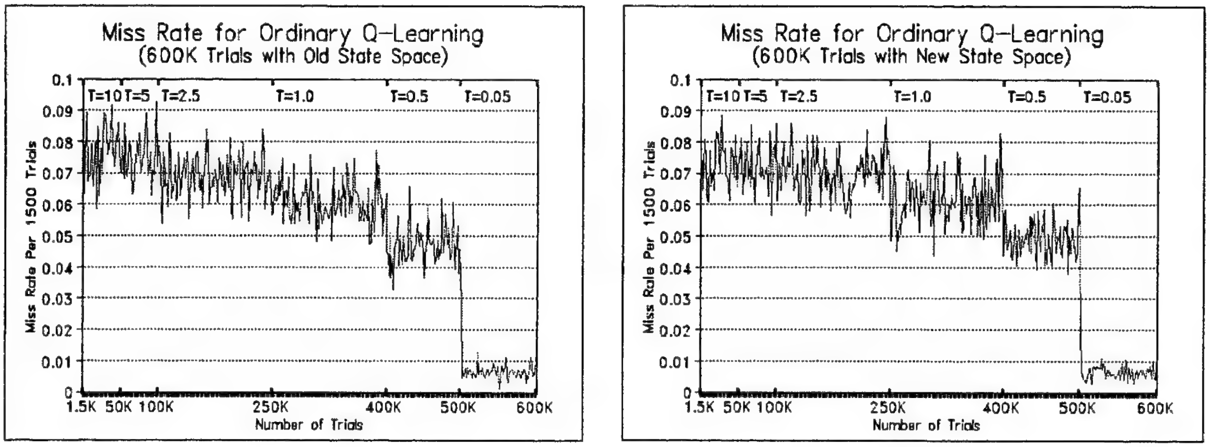
It is seen in Figs.20 and 21 that feature space performs well even with a small number of trials while frame space failed to give a satisfactory result due to insufficient exploration of its much larger number of state-action pairs. For comparison purpose, the experiments were performed again with a much larger number of trials (600,000). These computational intensive version of the experiment resulted in a slightly improved performance with the miss rate reduced from range of 0-0.02 to 0-0.01(Fig.22, 23). It is also shown from Fig.23 that feature space has a slightly poorer performance that that of the frame space , reflecting the fact that the generalized feature space failed to discriminate some of the distinct world states better interpreted in the more detailed frame space representation of perception-states..



(a)

(b)

Figure 22: Performance (average of total cost till target is detected from 10 trials at a interval of 1500 trials) by ordinary Q-learning for 600K trials with (a)frame space (b) feature space.



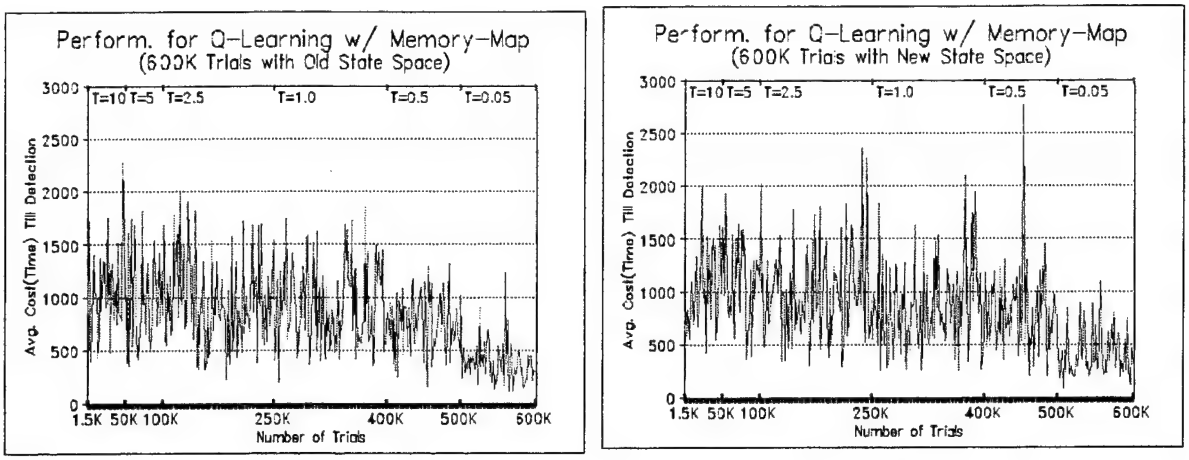
(a)

(b)

Figure 23: Miss rate (number of missed trials over the number of trials during the interval of 1500 trials) by ordinary Q-Learning for 600K trials with (a) frame space (b) feature space.

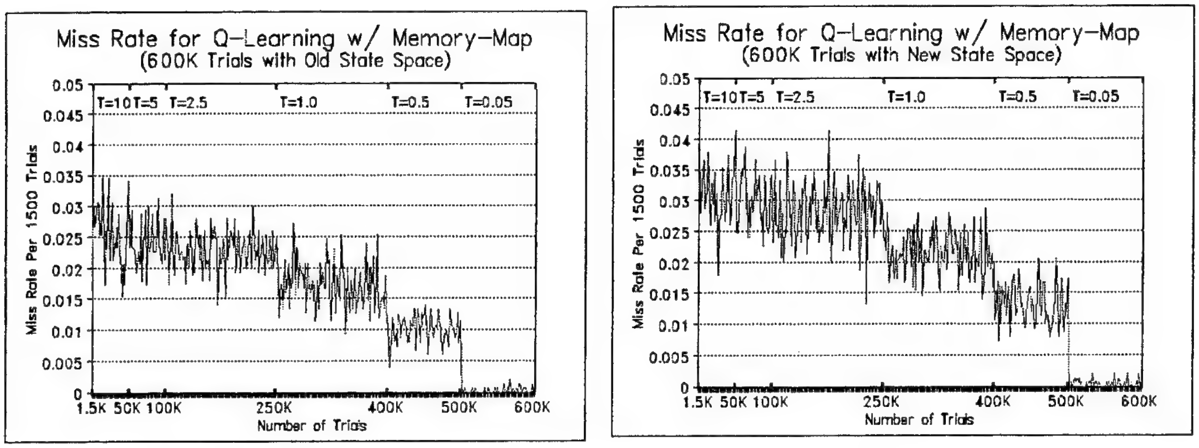
5.2 Q-Learning with fixation-window memory

The fixation-window approach described in Section 3.1.2 is used in the action-selection process to reduce learning convergence problems caused by the violation of the Markovian assumption. The fixation-window approach does not modify the learning algorithm, it just blocks the agent from inappropriate choices (visiting scene locations that has been visited before and which contain no target). The extended version (600K) of the experiment is initially performed:



(a) (b)

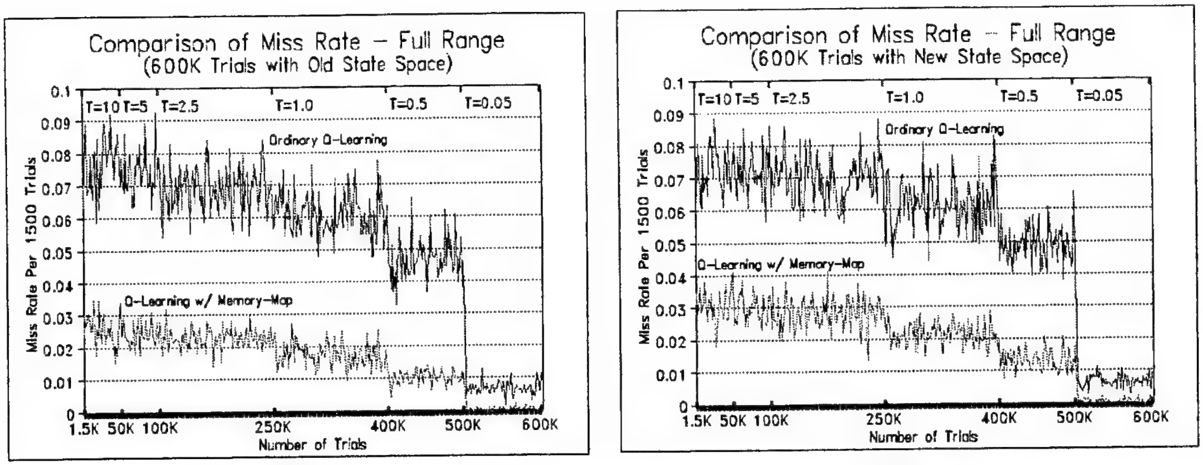
Figure 24: Performance (average of total cost till target is detected from 10 trials at a interval of 1500 trials) by Q-learning with fixation-window for 600K trials with (a)frame space (b) feature space.



(a) (b)

Figure 25: Miss rate using Q-Learning with fixation-window and (a) frame space (b) feature space

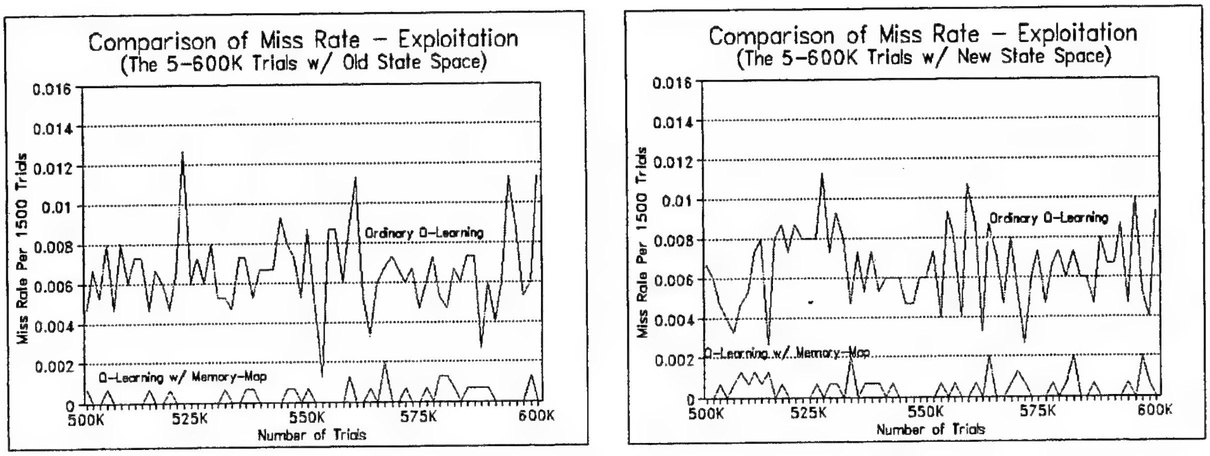
Comparing Figs. 22 and 24, the improvement in performance is not significant. However, comparing the miss rate indexes shown in Figs. 23 and .25, the fixation-window approach yields a greatly improved miss rate, even during the phase of random action selection (T=10). Fig. 26 demonstrates the overall improvement, and the comparison of the miss rate after the learning (T=0.05) in Fig. 27 show that the agent with the combination of Q-learning and fixation-window prerception-state can detect targets with a much lower miss rate (0.002) than the one with standard Q-leaning(0.01) in exploitation mode.



(a)

(b)

Figure 26: Comparison of the miss rate of 600K trials by standard Q-learning and 600K trials by Q-learning with fixation-window. Both (a) frame space and (b) feature space shown Q-learning with fixation-window has a lower miss rate than standard Q-learning.

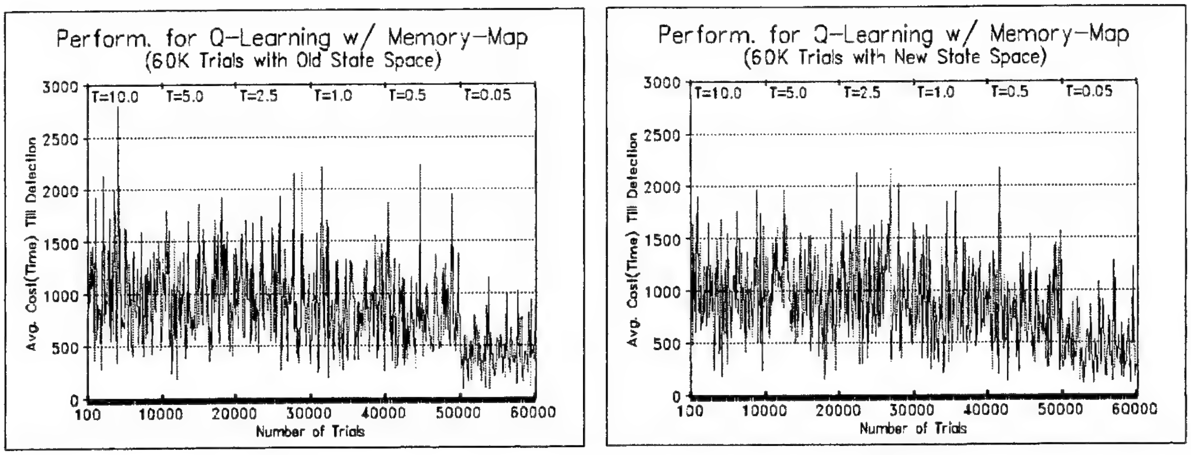


(a)

(b)

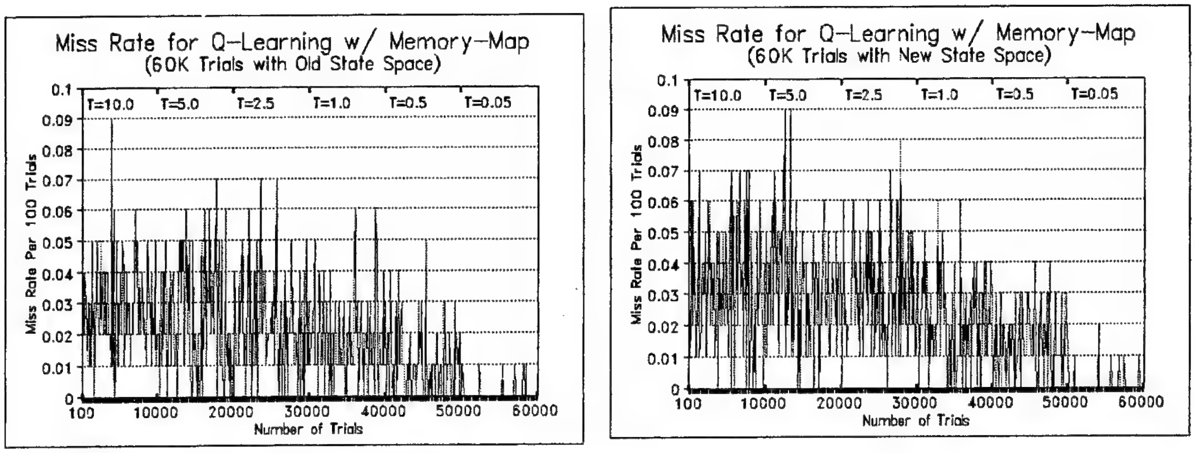
Figure 27: Comparison of the miss rate during exploration mode (T=0.05) from the 600K trials by standard Q-Learning and Q-Learning with fixation-window. Both (a) frame space and (b) feature space shown. Q-Learning with fixation-window has a lower miss rate.

The shorter version of the experiment(60,000 trials) is performed to test for consistency, and similar improvements from the fixation-window approach over the ordinary Q-learning are obtained as expected. The combination of Q-learning and memory-based approach is demonstrated to be an effective method for solving this non-Markovian POMDP control problem.



(a) (b)

Figure 28: Performance by Q-Learning with fixation-window and (a) frame space (b) feature space



(a) (b)

Figure 29: Miss rate by Q-Learning with fixation-window with (a) frame space (b) feature space-

6. Unsupervised learning for dimensionality reduction

The use of a multiresolution sensor described in Section 4.1 containing a 3x3 fovea but with a wider FOV than the 3x3 uniform resoluton sensor should have target detection benefits. This new sensor array has many more perception-states, however. A major computational obstacle which Richard Bellman labelled as the curse of dimensionality, is encountered during the process of implementing Q-leaning with a foveal sensor. We explored the use of clustering networks to reduce this problem. This section reviews our efforts, which were largely successful.

Conventional Q-learning assumes enumeration of a state-action cross-space look-up table. This is impractical with the of foveal sensor. Unrealistically large memory and training times are required.

The foveal sensor has 9 binary pixel values from the fovea, and each of its 8 outer ring rexels will produce one of the 10 possible real values(equivalently, the sum or the average of the 9 pixels values under its FOV). The construction of a discrete state space from these values will result in 5.12 x 10^{10} states. Clearly a lookup table with columns this large is infeasible..

It is commonly expected that in a large state space like this one in the expanded token problem, similar states should have similar Q-values and similar optimal actions. Some popular techniques for generalization over input space are investigated in order to find a compact state representation, we proposed a state estimator which can utilizes the raw sensor information as the input while output the associated enumerate state in a compact state space. The floowing section describes our attempts of using artificial neural network architecture and unsupervised learning framework to construct such a state estimator.

Two types of neural network architecture, namely Kohonen Network (KOSOM) and Adaptive Resonance Theory (ART) are investigated for their capabilities to generalize the raw sensor input data to compact state output.

6.1 Kohonen Network

Kohonen network (Kohonen 1989, 1990) became the subject of study because of its ability to discover significant regularities within the input data without external supervision. The Kohonen self-organizing feature map (KSOFM) partakes strongly of the biological inspiration to be found in the retinal cortex, it possesses the unique property that cluster centers of features aggregate geometrically within the network output layer. In other words, features that are similar by vitae of possessing a small Euclidean distance between them in feature space will stimulate responses in KSOFM output neurons that are also geometrically close to each other. It is hoped that the KSOFM can cluster the visual sensor input data into a more compact representation, and upon which a smaller state space for Q-Leaning can be constructed.

The Kohonen network architecture consists of two layers, an input layer and a output layer(Kohonen layer), and each of the input layer neuron has a feed-forward connection to each output layer neuron. The output layer (Kohonen layer) can be one-dimensional (Fig. 30) or two-dimensional (Fig. 31), and the output neuron is selected by using competitive learning - a process in which output layer neurons compete among themselves to acquire the ability to fire in response to given input patterns e.g. Winner-Takes-All. For KSOFM, the competitive learning can be accomplished in two ways: (a) maximum stimulus : (11)

$$WinningNeuron \Leftarrow (Neuron_k \text{ such that}) \sum_{i=1}^{n} w_{ik}x_i < \left(\sum_{i=1}^{n} w_{ij}x_j \right) \quad \text{for j=1,2,.......,n}$$

Where w_{ij} is the weight connecting from input neuron i to output neuron j, and x_i is the input signal to the input neuron i. The other approach would be (b) minimum Euclidean norm between the weight vector and input vector. Let $\boldsymbol{W}_j$ be the weight vector for output neuron j, and $\boldsymbol{x}$ be the input vector, n be the total number of output neurons:

:

$$WinningNeuron \Leftarrow Neuron_k \quad \text{where} \quad \|W_k - x\| < \|W_j - x\| \quad \text{for} \quad j=1,2,.....,n \tag{12}$$

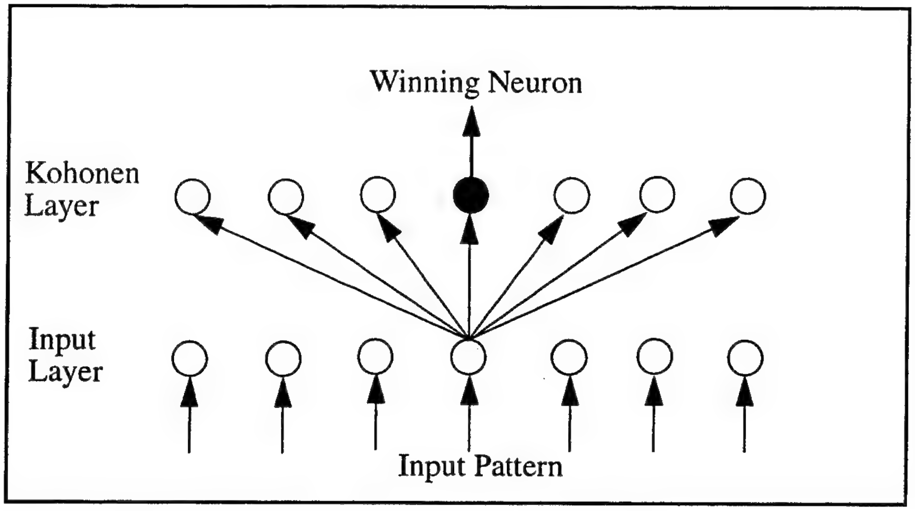


Figure 30: One-dimensional Kohonen network architecture

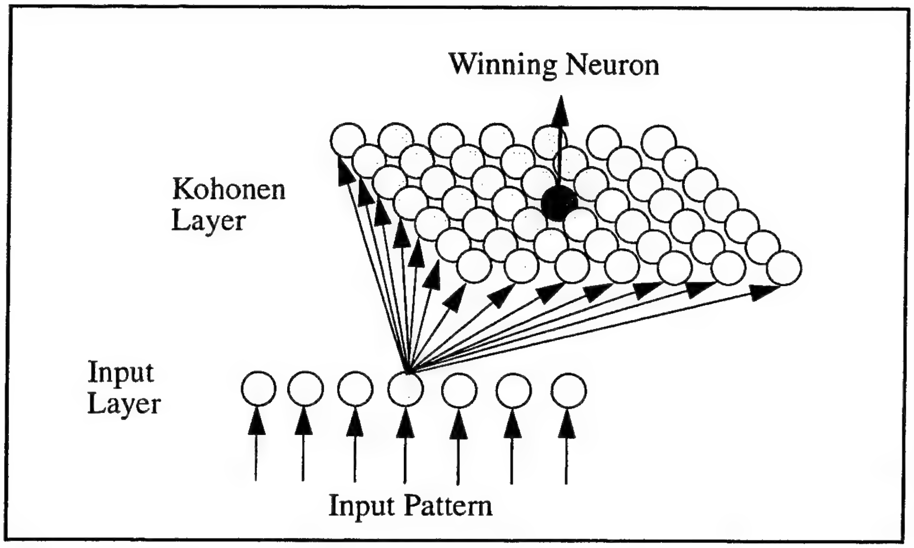


Figure 31: Two-Dimensional Kohonen network architecture

Kohonen SOFM is obtained by augmenting the Kohonen network with lateral feedback so that relatively like features are congregate within the output (Kohonen) layer. The lateral feedback is implemented by defining a neighborhood size controlling function surround the winning neuron. The size of the neighborhood is varying over the course of the network training, larger neighborhood means more positive feedback while the subsequent reduction of the neighborhood makes the clusters sharper so that the cluster response may be refined. The neighborhood function can be expressed as a rectangular or hexagonal lattics. (Fig. 32).

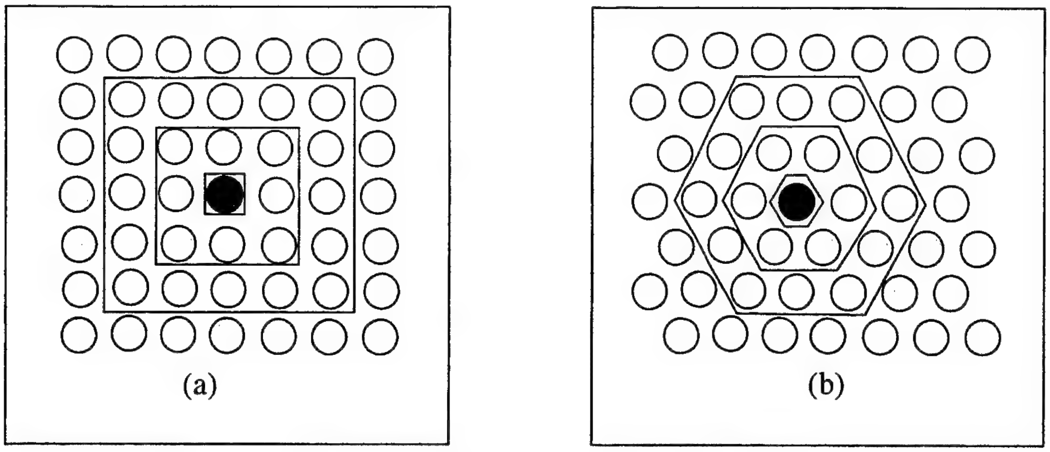


Figure 32: Neighborhoods on (a) rectangular lattics. (b) hexagonal lattice.

The weights of the connections are initially randomly assigned with small values and then updated according to the changing neighborhood function $A_{i(x)}(n)$ and the learning rate $\alpha(n)$, where n denotes the time steps and i(x) is the winning neuron on the output layer.

$$w_j(n+1) = \begin{cases} w_j(n) + \alpha(n)[x - w_j(n)] & j \in A_{i(x)}(n) \\ w_j(n) & \text{otherwise} \end{cases}$$

In our study, the Kohonen network used has 9 input neurons with respect to the 9 binary inputs value from the uniform-resolution sensor. There are initially 900 output neurons assigned to the 1-D Kohonen layer, each neuron represents a state thus it has its associated $S\,X\,A$ table to store the Q-values. The Kohonen network will be used as a State Estimator, sensor data will be feed as input while the winning output neuron implies the state for Q-learning. It is expected that after certain amount of training, the output patterns will at least converge to 512 Kohonen neurons with respect to the 512 distinct input pattern, and any reduction in the output numbers indicate the clustering of similar input pattern thus input generalization is achieved. The target detection experiment with Kohonen network is run for 2,000 trials, and the results are plotted in Fig. 33.

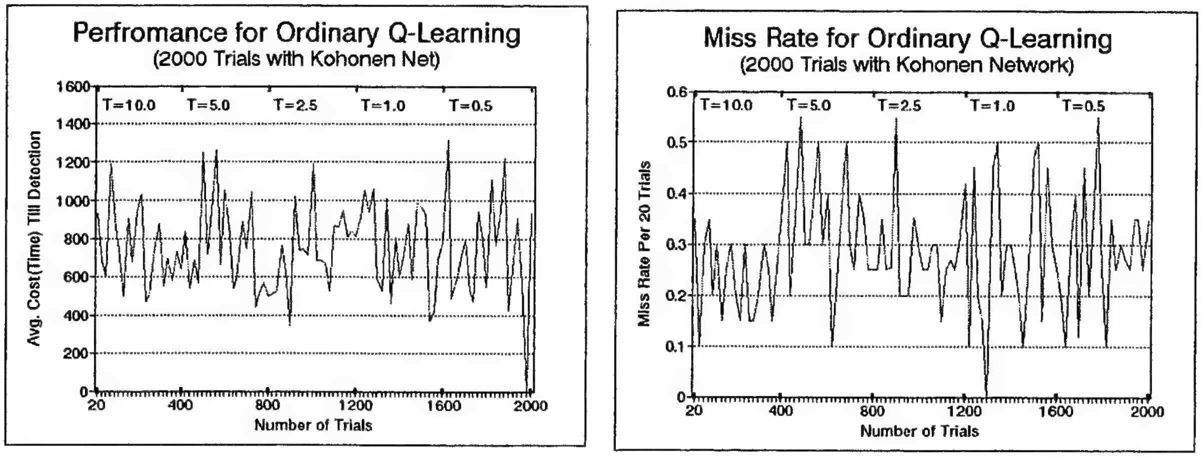


Figure 33: Performance and miss rate by Q-learning with Kohonen Network as state estimator for 2000 trials.

The performance and miss rate did not shown any improvements over the training - which implies the state allocation might not be stabilized yet. From the plot of the state utilization (Fig. 34), it is shown that all of the 900 states received about the same level of access, it is unable to determine if the Kohonen network is successful in clustering the input patterns. There are two explanations for the obtained results; 1) Due to the way that the environment is generated (IID pixels), the input pattern is not structured and randomly distributed in the feature space. This might create difficulties in clustering the patterns, or simply this is not the appropriate data pattern for efficient clustering. It is pointed out in (Kohonen 90) that raw pixel value are not fitted to be used as inputs to the Kohonen network, more structured features should be used. 2) 2000 trials may not be sufficient for the training of the Kohonen network, with some fine tuning in the weight updating algorithm (and maybe a reduction in the output space) and some extended training period, the resulted Kohonen network is most probable to generated a better performance in clustering. However, this also create a major difficulty, the computation involved in the Kohonen network is very intensive, it will take tremendous amount of time for this kind of extended training. Moreover, even if the resulted Kohonen state estimator is capable of generating compact state space, it will still take a lot more computations to generate a enumerate state from the input vector, this time-consuming mechanism will limit the performance of the algorithm in real-time or time-critical applications such as the target detection task.

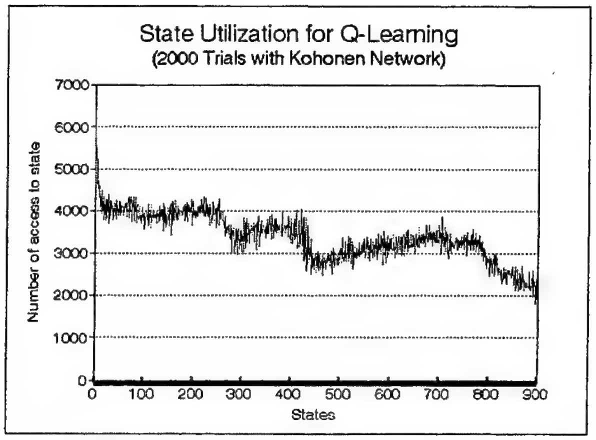


Figure 34: State utilization of the Kohonen state estimator.

6.2 Adaptive Resonance Theory (ART)

The ART (Carpenter and Grossberg 1987a) neural network architecture is considered for the building block of the state estimator because 1) ART network can classify input vectors according to the stored patterns, 2) It will adaptively create a new output neuron for the input vector if it cannot find a sufficiently similar stored pattern. 3) The ART2 (Carpenter and Grossberg 1987b) architecture modified from the ART network, is capable of handling both binary and continuous values(e.g. rexel values in the Foveal sensor).

The ART network consists of two layers and modifiable weights in both feed-forward and feedback directions(Fig. 35). The input pattern is feed to the Comparison layer (Input Layer) and through the feed-forward connections, a winning neuron is generated on the Recognition layer (output layer) by involving the winner-take-all paradigm on the stimuli. The weight values of the backward connections from the winning neuron to the input neurons actually stored the prototype of the associated pattern class. By comparing the associated backward weights and the original input pattern with a vigilance test, the winning neuron will be determined if it is the appropriate output for the input pattern.

The vigilance test provides a measure of the distance between the input vector and the cluster center corresponding to the firing recognition layer neuron, if the vigilance falls below a preset threshold, the winning neuron will be masked out from the rest of the competitions and a new round of feed-forward recognition then feed-backward comparison will be performed. The process will be repeated until the a winning neuron passed the vigilance test (correct classification), or when all the neurons in the recognition layer were tried. If all the neurons on the recognition layer fail the vigilance test, a new category will be created, the associated recognition layer neuron, feed-forward connection weights, as well as feed-backward prototype will be added to the network.

Whenever there is a new neuron added or a winning neuron passed the vigilance test, the winning neuron's connection weights are trained such that its associated cluster center in feature space is moved toward the input vector.

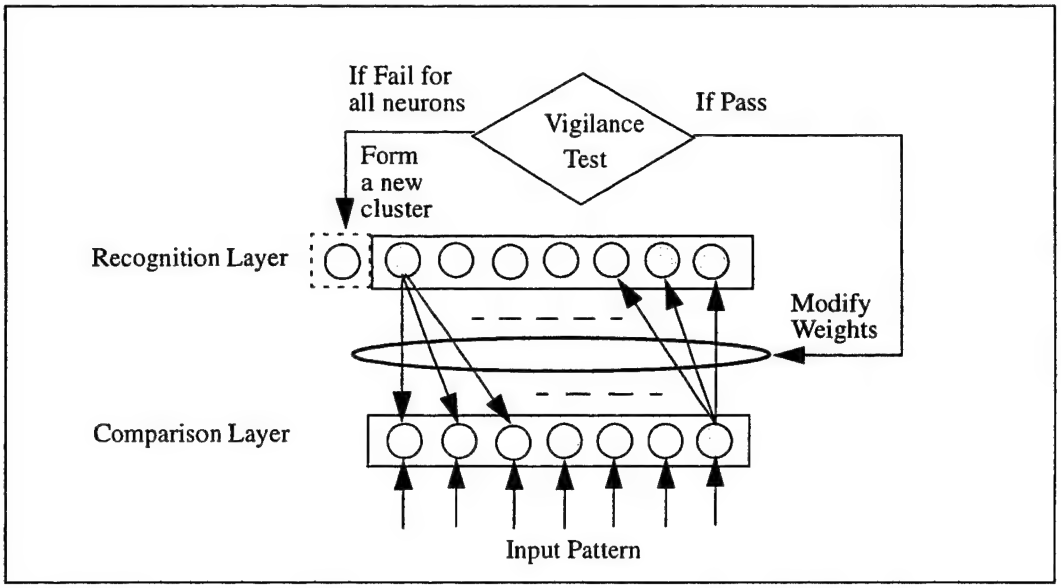


Figure 35: ART architecture

The vigilance test used in the ART with binary input values is usually the matching of binary input values and the prototype, then the matched percentage will be checked against the vigilance threshold for passing criteria. In our experiment of using ART to constructed a state estimator, there are 9 input neurons(corresponds to the 9 binary inputs in the uniform-resolution sensor) in the comparison layer and 1 output neurons (which will be augmented during the training process by the algorithm) in the recognition layer. A vigilance threshold of 0.8 is used, which indicates approximately 8 out of the 9 inputs will have to be matched.

The experiment of target detection is carried out for 2000 trials using ART to generate the states for Q-Learning, the performance and miss rate is plotted in Fig. 36. It is shown from the plotting that the performance is not improving much as the training goes by, this may due to the insufficient number of training (Only 2000 trials). However, this is exactly one of the major drawbacks of ART network and cannot be overcome; the process of using ART to generate a state from input data is too computationally intensive and consumes too much time during the process(It will perform the forward and backward calculation for all the output neurons in the worst case). Moreover, since the vigilance test does not require a fully match input-prototype pairs, the backward connection weights (the prototype) can be changed over time as many similar patterns passed by. This implies that the ART is forgetting the earliest patterns and slowly shifting the prototypes to the most recent seen patterns, thus while a earlier pattern is shown, a new category will be introduced. The "forgetting" behavior is verified from the experiment result that the number of neurons in the output layer is increased linearly with the number of trials and even exceed the expected maximum number of 512, the consequence of this increment is a even heavier computational and memory requirement.

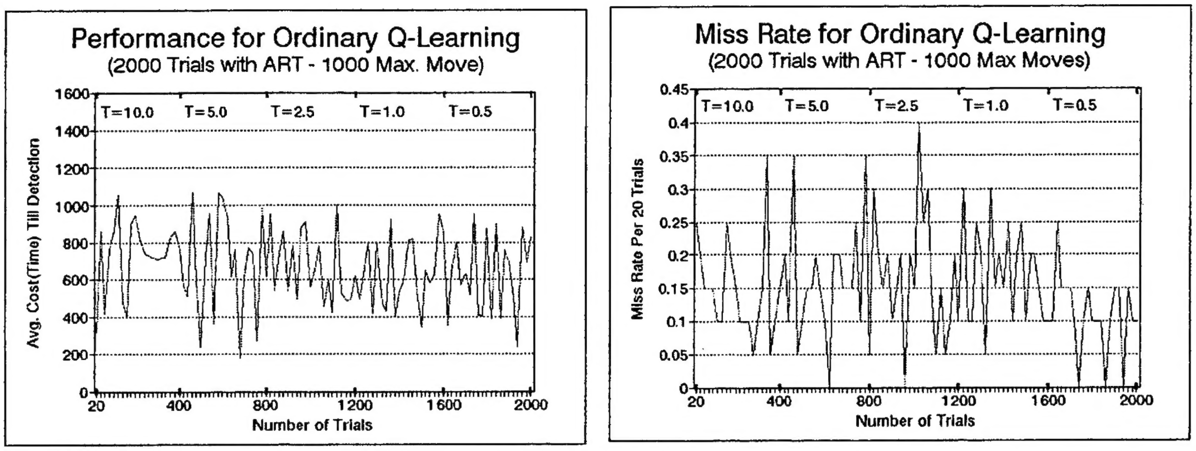


Figure 36: Performance and miss rate using Q-learning with ART as the state estimator

Weighing the results of the Kohonen and ART net studies, it appears that neither class of neural nets promotes useful clustering of states for the purpose of compact perception-state representations. The heuristics of feature space were a useful step in this direction but clearly must be better explored and understood as the problem is taken from its token form to more practical levels of computational and conceptual difficulty.

7. Conclusions and suggestions for further study

This report reviews progress in applying reinforcement learinng strategies based on Q-Learning to solve the target detection gaze control problem. The experiments were designed to test target detection performance in a reasonably difficult simulation environment: small target (9 pixels), low SNR (-12.2 dB), no prior knowledge of the number or locations of targets present, no structure to the background (white noise) to be exploited to eliminate regions. On the other hand, the target was static, two-dimensional, not distored or occluded by noise and the size and orientation of the target were known a priori. These additional features of realisitic problems remain to be explored.

. The curse of dimensionality precluded perfect cumulative perceptual memory retention needed to permit Markov full-state updating as is required for guaranteed convergence using Q-

Learning. Only the partially-observed or perception-state version of this learning regime is feasible even for the most constrained versions of the problem. Both past and current percept data were compressed to make for feasible state-action lookup table data structures.

Two features of past experience were explored as summaries of past experience: direction-register memory, in which only the direction to the previous fixation point was remembered, and fixation-window memory, in which the locations of all fixations within a space-bounded sliding window moving with the fixation sequence are remembered. These features were used to constrain the actions permissible, through the simple rule that having been determined not to be a target, a pixel location should not be revisited. Similarly, two representations of the current frame were employed as perception-states to reduce the number of states in the Q- or evaluation function lookup table: frame memory and feature memory. The former stored the previous percept just as observed, the latter summarized the previous percept into a smaller number of perception-states indicating the degrees and commpass orientations of partial matches with the target.

Table 4 summarizes the results. The unconstrained random policy, with no helpful heuristics or learning at all, is to move at random, selecting the action from among all actions in the action set (first row of Table 4). Using the current percept as perception-state for policy selection feedback results in a policy fully taking the current percept into account, but without any memory of the results of past observations, and is listed in the second row of Table 4 as Percept Q-Learning. As seen , this strategy cuts either distance or time costs roughly in half compared to the random-walk control policy. Adding fixation-window memory, that is, memory of the previous fixation points within a sliding window, and using the feature space representation of the current percept leads to the results shown in the final row of Table 4. This strategy, designated as Constrained Q-Learning, is cconstrained in action space by the no-repeated-visitation rule and in state space by the compression of the percept into partial-match features, and is referred to as Constrained Q-Learing. Percept Q-Learning is both more memory-intensive and has roughly 50% performance penalty (distance or time) relative to Constrained Q-Learning..

Table 4: Experiment Results with Action Set A & frame space

Results of 100 Trials (After 60K Trials of Training)	Distance to target (pixels)	Time to target (fixations)
Unconstrained random	1236.77	616.90
Percept Q-Learning	668.95	339.26
Constrained Q-Learning	412.08	206.56

Note that a carefully selected perception-state space (e.g. feature space) can result in improved performance together with less memory and computational requirements. One of the many possible feature space for this token problem maybe the Gabor transformation of the percept, wavelet basis functions containing information partitioned in both the spatial and frequency domains.

Our effort to compress the foveal sensor perception-state space to a computationally feassible dimension using unsupervised learning ANN's was not successful. However, ANNs are still under investigation for this purpose. Those with specially constructed feed-forward architectures requiring much less computation, and those using reinformcement-based error-backpropagation learning algorithms, may be able to solve the problem efficiently. The work of Shibata, et al. 1995 needs to be considered for this purpose. The look-up table representation of Q-values can also be compressed (with generalization) using a feedforward recurrent neural network as reported in Lin 1991.

Besides applying various generalization methods for dimensional reduction, there are other approaches, like multi-resolution state space, hierarchical control strategy and fuzzy logic etc. that may be considered. Due to the multi-resolution nature of the foveal sensor, it is quite reasonable and possible to construct a multi-resolution state space in corresponding to the input from the foveal sensor. The PartiGame algorithm reported in (Moore, 1994) demonstrated success in applying Q-leaning to a multi-resolution state space. A hierarchical control strategy (e.g. separate low-level controllers for different rings and a upper-level gating routine to decide the priority) also seems a promising approach for solving problems with multiple sensors(rings).

In summary, RLappears to be a promising approach to the difficult problem of gaze control from visual sensors. Q-Learning constrained to reduce memory and computational requirements proves feasible for a token problem in which the goal is target detection.

References

Asada, M., Noda, S., Taearatsumida, S., and Hosoda, K., (1995). Vision-Based Reinforcement Learning for Purposive Behavior Acquisition. *Proc. of IEEE Int. Conf. on Robotics and Automation.*

Asada, M., Noda, S., Taearatsumida, S., and Hosoda, K., (1996a). Behavior Acquisition via Vision-Based Robot Learning. *Robotics Research, The Seventh International Symosium.* Springer. pp.1314-1319.

Asada, M., Noda, S., Taearatsumida, S., and Hosoda, K., (1996b). Target Reaching Behavior Learning with Occlusion Detection and Avoidance for A Stereo Vision-Based Mobile Robot. *Proc. of ROBOLEARN96: An International Workshop on Learning for Autonomous Robots*

Bandera,C., (1990). *Foveal Machine Vision Systems*. Ph.D. dissertation, Department of Electrical and Computer Engineering, SUNY at Buffalo.

Bandera,C., Scott, P., (1989). Foveal Machine Vision System. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Cambridge, MA, November.

Bandera, C., and Scott, P.,(1996) *Fusion of sensors that interact dynamically for explorartory development of robust, fast object detection and recognition- AFOSR STTR Phase I Progress Review.* Amherst Systems and University at Buffalo.

Bellman, R., (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Carpenter, G.A., and Grossberg, S., (1987a) A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vision, Graphics Image Process.*, Vol 37, pp54-115

Carpenter, G.A., and Grossberg, S., (1987b) ART2 : self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, Vol 26, pp 4919-4930.

Fagg, A.h., Lotspeich, D., and Bekey, G.A., (1994). A Reinfrocement-Learning Approach to Reactive Control Policy Design for Autonomous Robots. *1994 IEEE Conference on Robotics and Automation.*

Kaelbling, L.O., Littman, M.L., and Moore, A.W.,(1996). Reinforcement Learning : A Survey. *Journal of Artificial Intteligence Research 4.* 237-285.

Kohonen, T., (1989) *Self-organizztion and Associative Memory,* 3rd ed., Springer-Verlag, Berlin.

Kohonen, T., (1990) The self-organizing map. *Proceedings of IEEE,* vol.78 no.9, 1464-1480.

Lin L-J., (1991) Programming robots using reinfocement learning and teaching. *Proceedings of AAAI-91* pp. 781-786.

Moore, A.W., (1994). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-apces. In Cowan, J.D., Tesauro, G., & Alspector, J. (eds), *Advances in Neural Information processing Systems 6,* pp.711-718 San Mateo, CA. Morgan Kaufman.

Pandya, A.D., Macy, Robert B., (1996) *Pattern Recognition with Neural Networks in C++,* IEEE Press.

Shibata, K., Nishimo, T., and Okabe, Y.,(1995). Active Perception Based on Reinforcement Learning. *Proccedings of WCNN'95.* Washington D.C., vol.2, pp.170-173.

Watkins, C.J.C.H., (1989). *Learning from Delayed Rewards.* Ph.D. thesis, King's College, Cambridge, UK.

Watkins, C.J.C.H., & Dayan, P.,(1992) Q-Learning. *Machine Learning,* 8(3), 279-292.

Whitehead, S.D., and Ballard, D.H., (1991) Learning to perceive and act by trial and error. *Machine Learning* 7, 45-83.

Zurada, J.M., (1992) *Introduction to Artificial Neural Systems.* West Publishing Company.